HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

Fraunhofer IIS

DIPLOMARBEIT

Herr
Enrico Goldhan

Abbilden eines Graphen auf die im Graphen identifizierten Grundstrukturen am Beispiel integrierter Schaltkreise

2011

Fakultät: **Mathematik/Naturwissenschaften/Informatik**

Diplomarbeit

Abbilden eines Graphen auf die im Graphen identifizierten Grundstrukturen am Beispiel integrierter Schaltkreise

Autor:
Enrico Goldhan

Studiengang:
Angewandte Mathematik

Seminargruppe:
MA06w1

Erstprüfer:
Prof. Dr. rer. nat. Peter Tittmann

Zweitprüfer:
Dipl. Inf. Andy Heinig

Mittweida, Oktober 2011

Bibliographische Angaben

Goldhan, Enrico:
Abbilden eines Graphen auf die im Graphen identifizierten Grundstrukturen am Beispiel integrierter Schaltkreise – Oktober 2011, 83 Seiten, 26 Abbildungen, 27 Tabellen.
Mittweida, Hochschule Mittweida (FH), University of Applied Sciences, Fakultät Mathematik/Naturwissenschaften/Informatik, Diplomarbeit, 2011

Referat:

Das Ziel der Diplomarbeit ist die Entwicklung eines Verfahrens, mit dem es möglich ist, große Graphen auf ähnliche Grundstrukturen zu untersuchen. Hierbei soll der Graph zunächst mit Hilfe spektraler Clusterverfahren in vergleichbare Teile partitioniert werden. Anschließend wird die Graph Edit-Distanz mit anderen Ähnlichkeitsmaßen auf ihre Tauglichkeit verglichen werden. Auf Grundlage der Graph Edit-Distanz soll ein Ähnlichkeitsmaß entwickelt werden, das entsprechende Grundstrukturen im Graphen erkennt. Das Verfahren wird im Folgenden angewandt, um innerhalb einer integrierten Schaltung Grundstrukturen zu finden.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen **3**

2.1 Graphentheorie 3

2.1.1 Definitionen 3

2.1.2 Graphenmatrizen 6

2.1.3 Operationen mit Graphen 8

2.2 Grundlagen aus der linearen Algebra 9

2.2.1 Eigenwerte und Eigenvektoren 9

2.2.2 Spektren von Graphen 11

3 Modellierung **14**

3.1 Integrierte Schaltkreise 14

3.2 Modellierung elektronischer Schaltkreise 15

3.3 Aufgabenstellung 16

4 Realisierung **20**

4.1 Partitionieren 20

4.1.1 Spektrale Bisektion 21

4.1.2 Spektrale Multisektion 27

4.2 Netzwerkzentralitäten 31

4.2.1 Knotengradzentralität 31

4.2.2 Eigenvektorzentralität 32

4.2.3 PageRank 33

4.2.4 Hub- & Authority-Gewichte 34

4.2.5 Die betweenness-Zentralität 35

4.3 Ähnlichkeit in Graphen 36

4.3.1 Maximum Common Subgraph (MCS) 37
4.3.2 Maß von Padadopoulos und Manolopoulos 38
4.3.3 Graph-Edit-Distanz 39
4.3.4 Näherungsalgorithmus 42
4.3.5 Einbettung der Graphen 45
4.4 Levenshteindistanz 46
4.5 Das zyklische String-Matching 47

5 Evaluierung der Ergebnisse **50**
5.1 Partitionierung des Graphen 50
5.2 Parameter der suboptimalen GED 54
5.2.1 Zentrumswahl *PC* und Ordnung der Nachbarschaft *PO* 54
5.2.2 Wahl der Kostenfunktion *PF* 56
5.2.3 Aufstellen der Ähnlichkeitsmatrix *PS* 57
5.3 Bestimmung der Prototypen und graphische Darstellung 58
5.4 Auswertung an den Testklassen 59
5.4.1 Vergleich der Parameter an der Testklasse 150 60
5.4.2 Testklasse 300 70
5.4.3 Testklasse 500 71
5.4.4 Testklasse 700 72
5.5 Bemerkung zur Implementierung 75

6 Zusammenfassung und Ausblick **77**

Literaturverzeichnis **80**

Abbildungsverzeichnis

2.1 Der ungerichtete Graph G. 5

3.1 Siliziumplättchen (Chip) 14
3.2 Phasen der Chipentwicklung 15
3.3 Elektrische Schaltung 15
3.4 Schlichter Graph 16
3.5 Verfahren zur Identifizierung ähnlicher Grundstrukturen 18
3.6 Cluster C, bestehend aus den Teilgraphen G, H, I, J und K mit zugehörigem Prototyp 19

4.1 Die Hypersphäre um den Hyperwürfel im n-dimensionalen Raum. . . 23
4.2 Der Umformungsweg besteht aus drei Kantenlöschungen, einer Knotenlöschung, einer Knoteneinfügung, zwei Kanteneinfügungen und zwei Knotensubstitutionen. 40
4.3 Stringerstellung mit Hilfe der Koordinaten der Knoten aus $N(u)$ im $\mathbb{R}^2$ 43
4.4 Beispiel für das Matching zweier Nachbarschaften $N(u)$ und $N(v)$, mit 4 substituierten Knoten, einem gelöschten und einem eingefügten Knoten 44

5.1 Spektrale Multisektion in zwei Teilgraphen ohne Größenvorgabe . . . 52
5.2 Vier verschiedene Einbettungen von G 55
5.3 Verhältnis der Reduktionsstufen zur Clusteranzahl 58
5.4 Der Graph T und der erzeugte Prototyp P 59
5.5 Prototyp des Clusters 5 aus GED_E_EM_CC2_D mit $RS = 3$ 62
5.6 Prototyp des Clusters 13 aus GED_E_EM_CC1_D mit $RS = 4$ 63
5.7 Prototyp des Clusters 12 aus GED_D_EM_CC1_D mit $RS = 4$ 64
5.8 Prototypen aus GED_E_C_CC1_D bzw. GED_E_DE_CC1_D mit $RS = 4$ 66
5.9 Prototyp des Clusters 30 aus GED_E_EM_CC1_P mit $RS = 2$ 68

5.10 Prototyp des Clusters 18 aus GED_E_EM_CC1_P mit $RS = 2$ 69
5.11 Prototyp des Clusters 10 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 300 71
5.12 Prototyp des Clusters 15 aus GED_E_EM_CC1_P mit $RS = 2$ der Testklasse 500 72
5.13 Prototyp des Clusters 4 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700 73
5.14 Prototyp des Clusters 5 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700 74
5.15 Prototyp des Clusters 9 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700 75

Tabellenverzeichnis

3.1 Ähnlichkeitsmatrix des Clusters C, samt Prototyp 19

4.1 Levenshteindistanz für Bachelor→Master 47
4.2 Der Umformungsweg für Bachelor→Master 47

5.1 Verlauf der spektralen Multisektion. 51
5.2 Entwicklung der Anzahl isolierter Knoten während der spektralen Bisektion 52
5.3 Größenverteilung innerhalb der Testklassen 53
5.4 Vergleich der Einbettungen mit *GED_E_C_CC1_D* 55
5.5 Vergleich verschiedener Kostenfuntkionen $CC1$ und $CC2$ an der Testklasse 150 61
5.6 Operationstabelle des Clusters 5 aus GED_E_EM_CC2_D mit $RS = 3$ 62
5.7 Operationstabelle des Clusters 13 aus GED_E_EM_CC1_D mit $RS = 4$ 62
5.8 Vergleich verschiedener Zentrumsbestimumngen an der Testklasse 150 63
5.9 Operationstabelle des Clusters 12 aus GED_D_EM_CC1_D mit $RS = 4$ 64
5.10 Vergleich verschiedener Nachbarschaftsanordnungen an der Testklasse 150 65
5.11 Operationstabelle des Clusters 3 aus GED_E_C_CC1_D mit $RS = 3$. 65
5.12 Operationstabelle des Clusters 1 aus GED_E_C_CC1_D mit $RS = 4$. 66
5.13 Operationstabelle des Clusters 3 aus GED_E_DE_CC1_D mit $RS = 4$ 66
5.14 Vergleich verschiedener Nachbarschaftsanordnungen an der Testklasse 150 67
5.15 Operationstabelle des Clusters 30 aus GED_E_EM_CC1_P mit $RS = 2$ 67
5.16 Operationstabelle des Clusters 18 aus GED_E_DE_CC1_P mit $RS = 2$ 68
5.17 Vergleich der GED für die Testklasse 300 70
5.18 Operationstabelle des Clusters 10 aus GED_E_EM_CC1_P mit $RS = 4$ der Klasse 300 70

5.19 Vergleich der GED für die Testklasse 500 71
5.20 Operationstabelle des Clusters 15 aus GED_E_EM_CC1_P mit $RS = 2$ der Testklasse 500 72
5.21 Vergleich der GED für die Testklasse 700 73
5.22 Operationstabelle des Clusters 4 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 7000 73
5.23 Operationstabelle des Clusters 5 aus GED_E_EM_CC1_P mit $RS = 2$ 74
5.24 Operationstabelle des Clusters 9 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700 74

1 Einleitung

Ein integrierter Schaltkreis (Integrated Circuit) ist eine auf einem einzelnen Substrat untergebrachte elektronische Schaltung. Das Hauptmerkmal der integrierten Schaltungen ist eine große Anzahl von verschiedenartigen oder gleichen Bauelementen, die miteinander verbunden werden. In den letzten 50 Jahren wurden die Arten und Anwendungen integrierter Schaltungen immer vielseitiger. Eine grobe Unterteilung sei hierbei durch Art der Fertigungstechnologie, der Signalart des Schaltkreises und dessen Anwendung gegeben.
Ein Integrated Circuit-Layout (IC-Layout) ist die Darstellung integrierter Schaltungen durch die Verwendung planarer geometrischer Formen für die verschiedenen Metall-, Halbleiter- und sonstigen Komponenten der integrierten Schaltung. Das Verhalten des endgültigen Schaltkreises hängt zu einem großen Teil vom Layout ab, d.h. von der Positionierung und Verbindung der geometrischen Formen. Ein gutes Layout steigert hierbei die Leistungsfähigkeit eines Chips, vermindert seine Größe oder sorgt erst dafür, dass dieser Chip überhaupt hergestellt werden kann [Alb07]. Mittlerweile sind die Layouts der Schaltungen zu groß, um sie noch manuell bearbeiten zu können. Daher kommen vor allem von Algorithmen gesteuerte Layoutverfahren zum Einsatz, deren Verlauf und Ergebnisse für den Menschen kaum nach zu vollziehen sind. Das Ziel der Diplomarbeit ist es, ein Verfahren zu entwickeln, mit dem es möglich ist, die Chiplayouts als Graphen zu modellieren und in diesen großen Graphen ähnliche Grundstrukturen, d.h. Komponenten, zu identifizieren.
Dadurch soll die zukünftige Layoutentwicklung vereinfacht werden, indem man innerhalb eines Layouts ähnliche Grundstrukturen erkennen und markieren kann, wodurch das gesamte Layout übersichtlicher gestaltet wird. Um das IC-Layout auf ähnliche Grundstrukturen untersuchen zu können, werden Methoden aus den mathematischen Gebieten der Graphentheorie, der linearen Algebra und Verfahren aus der Datenanalyse genutzt.

Daher werden in Kapitel 2 Grundlagen aus den Bereichen der Graphentheorie und der linearen Algebra vermittelt, die zur Bearbeitung der Problematik nötig sind.
Kapitel 3 enthält eine kurze Einführung in die Entwicklung neuer integrierter Schaltungen, sowie die Modellierung der vorliegenden Aufgabenstellung.
In Kapitel 4 werden die angewendeten Verfahren und die Grundideen hinter ihnen vorgestellt. Dabei werden Algorithmen der spektralen Graphenpartitionierung, Netzwerkzentralitäten und Graphenähnlichkeitsmaße präsentiert. Vor allem auf die Graph Edit-Distanz und ihre suboptimale Berechnung soll hierbei näher eingegangen werden.
Kapitel 5 umfasst die Auswertung der Ergebnisse. Hierbei werden neben den Methoden der spektralen Partitionierung auch die verschiedenen Verfahren der suboptimalen Berechnung der Graph Edit-Distanz mit Hilfe verschiedener Netzwerkzentralitäten auf ihre Tauglichkeit überprüft und miteinander verglichen.
Kapitel 6 enthält die Zusammenfassung der Diplomarbeit. Die aus der Diplomarbeit gewonnenen Erkenntnisse werden präsentiert, sowie ein kurzer Ausblick auf eine mögliche Weiterführung zur Lösung der vorgestellten Problematik gegeben.

2 Grundlagen

2.1 Graphentheorie

In diesem Abschnitt soll eine kurze Einführung in das Thema der Graphentheorie gegeben werden. Die Definitionen seien hierbei an [Tit03], [BE05] und [Neu75] angelehnt.
Graphen sind mathematische Modelle für netzartige Strukturen in Natur und Technik. Dazu zählen neben elektrischen Schaltungen auch Straßennetze, chemische Moleküle oder wirtschaftliche Verflechtungsbeziehungen. Alle diese Netze besitzen eine gemeinsame Grundeigenschaft. Sie bestehen aus zwei verschiedenartigen Mengen von Objekten. Die Objekte erster Art sind zum Beispiel Orte im Straßennetz oder Bauteile eines Schaltkreises. Sie werden durch Objekte der zweiten Art verbunden. Das sind zum Beispiel Straßen oder Übertragungsleitungen. In der Terminologie der Graphentheorie werden diese Objekte als Knoten und Kanten bezeichnet. Die Graphentheorie untersucht zunächst nur die rein strukturellen Fragen einer Netzstruktur. Ein Graph ist nur allein durch die Menge seiner Knoten und seiner Kanten sowie durch die Zuordnung der Endknoten einer Kante bestimmt. Damit gehen in diesem abstrakten Modell alle Informationen über die konkrete Art und Beschaffenheit der Knoten und Kanten verloren. Jedoch verbleiben viele Eigenschaften eines Netzes, die bereits auf dieser Abstraktionsstufe untersucht werden können. Die Untersuchung liefert die Grundlage für die Lösung kombinatorischer Optimierungsprobleme auf Graphen und für die Konstruktion von Algorithmen zum Entwurf und zur Analyse von Netzstrukturen [Tit03].

2.1.1 Definitionen

Definition 2.1
Ein **ungerichteter Graph** $G = (V, E)$ besteht aus einer **Kontenmenge** V und

einer **Kantenmenge** $E \in V \times V$, wobei jeder **Kante** $e \in E$ von G zwei (nicht notwendig verschiedene) **Knoten** aus V zugeordnet sind.

Bemerkung:
Die Anzahl der Knoten und Kanten wird mit n beziehungsweise m bezeichnet. Kanten werden in der Form $e = \{u, v\}$ beschrieben. Hierbei sind u und v die **Endknoten** der Kante e. Wenn v ein Endknoten der Kante e ist, so heißt v **inzident** zu e.

Definition 2.2
Ein **gerichteter Graph** $G = (V, E)$ besteht aus einer **Knotenmenge** V und einer **Bogenmenge** E, sodass jedem **Bogen** (jeder **gerichteten Kante**) $e = (u, v)$ ein eindeutig *geordnetes* Paar (u, v) von Knoten aus V zugeordnet ist.

Bemerkung:
Der Knoten u heißt der **Anfangsknoten**, v der **Endknoten** des Bogens $e = (u, v)$. Um die Ordnung des Knotenpaares hervorzuheben, schreibt man (u, v) statt $\{u, v\}$. Eine Kante der Form $e = \{v, v\}$, für welche der Start- und Endknoten zusammenfallen, heißt **Schlinge**. Zwei Kanten $e = \{u, v\}$ und $f = \{u, v\}$ zwischen denselben Endknoten heißen **parallel**.

Definition 2.3
Ein Graph ohne Schlingen und parallele Kanten heißt **schlichter Graph**. Im Folgenden werden nur schlichte Graphen betrachtet, die kurz als Graphen bezeichnet werden.

Bemerkung:
Die Zugehörigkeit einer Knotenmenge V und einer Kantenmenge E zu einem Graphen G wird durch die Schreibweise $V(G)$ bzw. $E(G)$ verdeutlicht.
Um einen Graphen bildlich darzustellen, werden Knoten durch Punkte oder kleine Kreise präsentiert. Eine Kante wird durch eine Strecke oder eine Kurve, die zwei Knoten miteinander verbindet, dargestellt. Der in Abb. 2.1 dargestellte Graph ergibt sich für G=({1,2,3,4}{a,b,c,d,e}) mit $a = \{1, 2\}$, $b = \{1, 3\}$, $c = \{2, 3\}$, $d = \{2, 4\}$, $e = \{3, 4\}$.

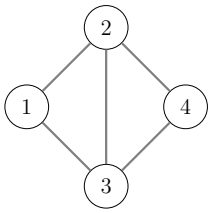

Abb. 2.1: Der ungerichtete Graph G.

Definition 2.4

Zwei Knoten $u, v \in V(G)$, die durch die Kante $e = \{u, v\}$ verbunden sind, werden als **adjazent** oder **benachbart** bezeichnet. Die Menge $\Gamma(v)$ aller zu einem Knoten $v \in V$ adjazenten Knoten wird als **Nachbarschaft** von v bezeichnet. Der **Grad** $\deg(v)$ eines Knoten $v \in V(G)$ ist die Anzahl der von v ausgehenden Kanten in G. Ein Knoten mit Grad null wird als **isolierter Knoten** bezeichnet. Für schlichte Graphen gilt $\deg(v) = |\Gamma(v)|$.

Der **durchschnittliche Knotengrad** $\vartheta(G)$ resultiert aus dem Durchschnitt aller Knotengrade von G.

$$\vartheta(G) = \frac{1}{n} \sum_{i=1}^{n} \deg(v_i) \tag{2.1}$$

Definition 2.5

Eine **Kantenfolge** ist eine Folge $v_1, e_1, v_2, e_2, ..., v_{k-1}, e_{k.1}, v_k$ von Knoten und Kanten eines Graphen G, sodass die Kante e_i für $i = 1, ..., k-1$ jeweils die Endknoten v_i und v_{i+1} besitzt. Diese Kantenfolge **verbindet** v_1 mit v_k. Eine Kantenfolge ist ein **Weg**, wenn jeder Knoten aus V höchstens einmal in dieser Folge auftritt, wodurch auch keine Kante doppelt vorkommen kann.

Definition 2.6

Der **Abstand** $d(u, v)$ zweier Knoten $u \in V$ und $v \in V$ eines Graphen ist die Länge eines kürzesten Weges von u nach v. Existiert kein solcher Weg, d.h. liegen u und v in verschiedenen Komponenten von G, ist $d(u, v) = \infty$. Der Abstand eines Knoten zu sich selbst ist stets null. Die **Exzentrizität** eines Knotens $v \in V$ ist definiert als

$$e(v) = \max\{d(u, v) : u \in V\}.$$

Damit ergibt sich der **Radius** $r(G)$ als

$$r(G) = \min\{e(v) : v \in V\}.$$

Ein Knoten $v \in V$ liegt im **Zentrum** des Graphen, wenn $e(v) = r(G)$. Zur Berechnung der kürzesten Wege innerhalb eines Graphen kann der Algorithmus von Dijkstra verwendet werden [KK89].

Definition 2.7
Ein Graph $H = (W, F)$ ist ein **Untergraph** des Graphen $G = (V, E)$, wenn $W \subset V$ und $F \subset E$ gilt. Der **Nachbarschaftsuntergraph** $N(v)$ eines Knoten v des Graphen G besteht aus der Knotenmenge $W = \{v, u_i\}$, $u_i \in \Gamma(v)$, die v und dessen Nachbarschaft enthält und der Kantenmenge $F = \{f_i\}$, $f_i = \{w_i, w_j\}$, $w_i, w_j \in W$, welche die Kanten zwischen diesen Knoten enthält.

Definition 2.8
Ein Graph $G = (V, E)$ heißt **zusammenhängend**, wenn zwischen je zwei Knoten u und v seiner Knotenmenge ein Weg existiert. Ein zusammenhängender Untergraph eines Graphen G heißt eine **Komponente** von G.

Definition 2.9
Eine Abbildung $\phi : X \to Y$ heißt **bijektiv**, wenn $\forall y \in Y$ genau ein $x \in X$ existiert, so dass $\phi(x) = y$.
Zwei Graphen $G = (V, E)$ und $H = (W, F)$ heißen **isomorph**, wenn eine bijektive Abbildung $\phi : V \to W$ existiert, so dass für alle $v, w \in V$ die Anzahl der Kanten zwischen $\phi(v)$ und $\phi(w)$ in H ist. H geht damit aus einer Umbenennung der Knoten und Kanten von G hervor. Dann gilt $G \cong H$.

2.1.2 Graphenmatrizen

Definition 2.10
Die Darstellung eines Graphen G folgt meist in Form seiner **Adjazenzmatrix** $A(G)$. Sei G ein schlichter Graph mit n Knoten, dann ist $A(G)$ eine $n \times n$ Matrix mit den

Einträgen

$$a_{i,j} = \begin{cases} 1, & \text{i und j sind adjazent,} \\ 0, & \text{sonst.} \end{cases} . \tag{2.2}$$

Die Adjazenzmatrix des Beispielgraphen G aus Abb. 2.1 lautet

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.3}$$

Bemerkung:

Die Adjazenzmatrix eines ungerichteten Graphen ist stets symmetrisch, d.h. $a_{i,j} = a_{j,i}$. Die Summe der Elemente einer Spalte oder Zeile ist gleich dem Knotengrad:

$$\sum_{i=1}^{n} a_{i,k} = \sum_{j=1}^{n} a_{k,j} = \deg(k) \tag{2.4}$$

Definition 2.11

Eine weitere Graphmatrix ist die **Gradmatrix** D des Graphen G. Die Gradmatrix ist eine $n \times n$-Diagonalmatrix, deren Diagonalelemente die Knotengrade von G sind, d.h. $d_i = \sum_{j=1}^{n} a_{i,j}$. Für den Beispielgraphen G ergibt sich die Gradmatrix

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{2.5}$$

Definition 2.12

Damit lässt sich die **Laplacematrix** eines Graphen als Differenz aus der Gradmatrix und der Adjazenzmatrix definieren:

$$L = D - A \tag{2.6}$$

2.1.3 Operationen mit Graphen

Definition 2.13

Es sei $G = (V, E)$ ein Graph. Das **Entfernen einer Kante** $e \in E$ erzeugt aus G einen neuen Graphen $G - e = (V, E \backslash \{e\})$. Wenn $F \in E$ eine Kantenteilmenge des Graphen $G = (V, E)$ ist, so sei $G - F$ der Graph, der aus G durch entfernen aller Kanten aus F hervorgeht. Analog lässt sich das **Hinzufügen von Kanten** für schlichte Graphen definieren. Hierbei sei $G + e = (V, E \cup \{e\}))$ der neue Graph, der durch hinzufügen der Kante $e \notin E$ entsteht.

Definition 2.14

Für einen Knoten $v \in V$ wird $G - v$ als der Graph definiert, der aus G durch **Entfernen des Knotens** v hervorgeht. Dies schließt gleichzeitig das Entfernen aller zu v inzidenten Kanten des Graphen ein. Für $X \subset V$ ist $G - X$ der Graph, der durch entfernen aller Knoten aus X aus dem Graphen G hervorgeht. Das Hinzufügen von Knoten $v \notin V$ sei analog definiert.

Definition 2.15

Es sei $G = (V, E)$ ein zusammenhängender Graph. Eine Kantenteilmenge $F \subset E$ ist eine **Schnittmenge** von G, wenn $G - F = (V, E \backslash F)$ ein nicht zusammenhängender Graph ist. Eine Schnittmenge F heißt **minimal**, wenn keine echte Teilmenge von F ebenfalls eine Schnittmenge von G ist. Jede Menge D mit $F \subset D$ ist ebenfalls eine Schnittmenge von G.

Definition 2.16

Sei $G = (V, E)$ ein ungerichteter Graph. Eine **Partitionierung** P von G ist eine disjunkte Zerlegung der Knotenmenge V in k nichtleere Teilmengen $V_1, V_2, ..., V_k$, für die gilt:

$$P = \{V_1, V_2, ..., V_k\}$$

$$V = \bigcup_{i=1}^{k} V_i$$

$$\forall i, j = 1, ..., k,\ i \neq j : V_i \cap V_j = \emptyset.$$

Die Mengen V_i werden im Folgenden als *Block* und die aus den Teilmengen von

$V(G)$ induzierten Graphen als *Teilgraphen* $G_i = (V_i, E_i)$ von G bezeichnet. Die Partitionierung eines Graphen in $k = 1$ oder $k = n$ Blöcke ist trivial. Eine Zerlegung mit $k = 2$ wird als **Bisektion** des Graphen bezeichnet.

Definition 2.17
Sei $G = (V, E)$ ein ungerichteter Graph mit der Partition $P = \{V_1, ..., V_k\}$. Der **Schnitt** der Partition ist die Menge $S(P)$, die alle Kanten enthält, deren Endknoten sich in verschiedenen Blöcken von P befinden:

$$S(P) = \{e = \{u, v\} \in E(G) | v \in V_i, u \in V_j, i \neq j\}.$$

Die **Größe** des Schnitts ist gleich der Anzahl der Kanten in $S(P)$, d.h. gleich der Mächtigkeit von $S(P)$.

2.2 Grundlagen aus der linearen Algebra

Der folgende Abschnitt soll Grundlagen zur Berechnung von Eigenwerten und Eigenvektoren von Matrizen aus der linearen Algebra vermitteln. Die Eigenwerte und -vektoren der Laplacematrix eines Graphen werden von spektralen Verfahren genutzt [4.1.1] , um strukturelle Eigenschaften des Graphen zu charakterisieren. Die Definitionen sind [BE05, Seite 372] entnommen.

2.2.1 Eigenwerte und Eigenvektoren

Definition 2.18
Sei $M = (m_{i,j}) \in \mathbb{C}^{n \times n}$ eine $n \times n$ Matrix mit komplexen Zahlen als Einträgen. Ein Vektor $\vec{x} \in \mathbb{C}^n$ ist ein **Eigenvektor** mit dem **Eigenwert** $\lambda \in \mathbb{C}$ von M, wenn $\vec{x}$ und λ folgende Gleichung erfüllen.

$$M\vec{x} = \lambda\vec{x}, \text{ mit } \vec{x} \neq \vec{0}. \tag{2.7}$$

Bemerkung:
Der Nullvektor $\vec{0}$ wird als triviale Lösung nicht weiter betrachtet, da jedes $\lambda \in \mathbb{C}$ eine Lösung von $M\vec{0} = \lambda\vec{0}$ ist. Die Eigenwertgleichung (2.7) hat nur dann eine Lösung,

wenn gilt $\det(M - \lambda I) = \vec{0}$.

Definition 2.19

Das **Spektrum** von M ist definiert als die Multimenge aller Eigenwerte λ_n von M, wobei die Vielfachheit eines Eigenwertes λ_i in der Multimenge gleich seiner algebraischen Vielfachheit als Wurzel von p_M entspricht.
Sei M nun eine symmetrische Matrix mit reell wertigen Einträgen. Es existiert eine reguläre Matrix Q, so dass $M' = Q^{-1}MQ$ eine Diagonalmatrix ist und $Q^{-1} = Q^T$ gilt. Für den Beweis siehe [BE05].

Bemerkung:

Die Einheitsvektoren e_i des $\mathbb{R}^n$ sind Eigenvektoren von M', wobei die Eigenwerte $\lambda_i := m'_{i,i}$ die Diagonalelemente in der i-ten Spalten bzw. Zeile sind. Da das charakteristische Polynom unverändert geblieben ist, gilt:

$$\det(M - \vec{\lambda}I) = \det(M' - \vec{\lambda}I) = \prod_{i=1}^{n}(\lambda_i - \lambda). \tag{2.8}$$

Das Spektrum von M besteht aus n nicht zwangsweise verschiedenen Eigenwerten $\lambda_1 \leq ... \leq \lambda_n \in \mathbb{R}$.

Theorem 2.1. *Sei $M \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix, dann gilt:*

1. *Die Eigenwerte von M sind reell wertig und n Eigenvektoren bilden eine Orthonormalbasis im $\mathbb{R}^n$.*

2. *Ein Eigenwert λ_i von A mit $\lambda_i = \lambda_{i+1} = ... = \lambda_{i+k-1}$ hat die Vielfachheit k. Daraus folgt, dass im $\mathbb{R}^n$ genau k linear unabhängige Eigenvektoren zum Eigenwert λ_i existieren.*

3. *Es existiert eine orthogonale Matrix $Q = \begin{pmatrix} | & | & & | \\ \vec{y}_1 & \vec{y}_2 & ... & \vec{y}_n \\ | & | & & | \end{pmatrix}$ für die gilt*

$$Q^T MQ = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}.$$

4. $det(M) = \prod_{i=1}^{n} \lambda_i$.

2.2.2 Spektren von Graphen

Theorem 2.2. *Für die Laplacematrix $L = D - A$ eines Graphen G gilt:*

1. *Für jeden Vektor $\vec{f} \in \mathbb{R}^n$ gilt: $\vec{f}^T L \vec{f} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}(\vec{f}_i - \vec{f}_j)^2$.*
2. *L ist symmetrisch und positiv semidefinit, d.h. L besitzt n nichtnegative reellwertige Eigenwerte $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.*
3. *Der kleinste Eigenwert von L ist 0, der zugehörige Eigenvektor der konstante Einsvektor.*

Beweis:

1. Aus der Definition von $d_i = \sum_{j=1}^{n} a_{i,j}$ folgt:
   $\vec{f}^T L \vec{f} = \vec{f}^T D \vec{f} - \vec{f}^T A \vec{f} = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} f_i f_j$
   $= \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} f_i f_j + \sum_{j=1}^{n} d_j f_j^2) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}(f_i - f_j)^2$.
2. + 3. folgt aus der Symmetrie von A und D. Die positive Semidefinitheit folgt aus Eigenschaft 1., da $\vec{f}^T L \vec{f} \geq 0 \, \forall \vec{f} \in \mathbb{R}^n$.

Bemerkung:
Es existiert ein weiterer Weg, die Laplacematrix eines ungerichteten Graphen zu definieren. Sei σ eine beliebige **Orientierung** von G, d.h. jeder Kante $e = (u, v)$ wird eine Richtung gegeben, indem u oder v als Start- bzw. Endknoten ausgewählt wird. Die Einträge der Inzidenzmatrix $B = (b_{u,e})$ des gerichteten Graphen (G, σ) seien wie folgt definiert:

$$b_{u,e} = \begin{cases} -1, & u \text{ ist Startknoten von } e, \\ 1, & u \text{ ist Endknoten von } e, \\ 0, & \text{sonst.} \end{cases}$$

Unabhängig von der Wahl der Orientierung σ gilt $L = BB^T$. Für den zugehörigen Beweis siehe [BE05, Seite 380].

Lemma 2.1. *Für alle* $\vec{x} \in \mathbb{C}^n$, $\vec{x}^T L\vec{x} = \vec{x}^T BB^T\vec{x} = \sum_{\{u,v\}\in E} (x_u - x_v)^2$.

Lemma 2.2. *Ein Graph* $G = (V, E)$ *besteht aus* k *zusammenhängenden Komponenten, wenn gilt:* $\lambda_1(L) = ... = \lambda_k(L) = 0$ *und* $\lambda_{k+1}(L) > 0$.

Beweis:
Der Beweis ist [BE05, Seite 380] entnommen. Es sei B die Inzidenzmatrix einer beliebigen Orientierung von G. Für jede Komponente C von G gilt für den **Indikatorvektor** $\vec{s}(C) \in \mathbb{R}^n$

$$s(C)_i = \begin{cases} 1, & u \in V(C), \\ 0, & \text{sonst.} \end{cases}$$

Dann heißt $S := \{\vec{s(C)} | C \text{ ist Komponente von } G\}$ linear unabhängig und $L\vec{s}(C) = BB^T\vec{s}(C) = \vec{0}$. Jede zusammenhängende Komponente kann injektiv auf eine linear unabhängige Menge von Eigenvektoren mit dem Eigenwert 0 abgebildet werden. Gilt weiterhin das $\vec{s} \in \mathbb{R}^n$ ein Vektor ist, so dass $L\vec{s} = BB^T\vec{s} = \vec{0}$, dann folgt aus $\vec{s}^T BB^T\vec{s} = 0$, dass $B^T\vec{s} = \vec{0}$. Das bedeute, dass $\vec{s}$ konstant auf jeder zusammenhängenden Komponente sein muss. $\vec{s}$ bezeichnet eine Linearkombination der Elemente aus S. Die Anzahl der Komponenten ist gleich der Anzahl der linear unabhängigen Eigenvektoren mit Eigenwert 0.

Bemerkung:
Die Knoten seien gemäß ihrer zugehörigen Komponenten in der Adjazenzmatrix A angeordnet. Daraus ergibt sich für A eine Blockdiagonalform, die sich ebenfalls für die dazugehörige Laplacematrix L ergibt.

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}$$

Jede Teilmatrix L_i bezeichnet hierbei die Laplacematrix des zugehörigen Teilgraphen der i-ten Komponente. Das Spektrum der Blockdiagonalmatrix ergibt sich als Vereinigung der Spektren der einzelnen Blöcke und die dazugehörigen Eigenvektoren von L sind gleich den Eigenvektoren von L_i.

3 Modellierung

3.1 Integrierte Schaltkreise

In der heutigen Zeit wird der Alltag des Menschen zunehmend von neuester Technik bestimmt und unterstützt. Ob es die programmierbare Kaffeemaschine oder der Routenplaner in einem modernen Auto sind, all diese Bequemlichkeiten werden erst durch die Nutzung von Chips ermöglicht. Ein Chip ist ein Träger, auf dem eine elektronische Schaltung aufgebracht wurde. Erst durch sie können vorgegebene Prozesse durch elektrische Signale gesteuert und geregelt werden. Die aufgebrachte elektronische Schaltung wird auch als integrierte Schaltung bzw. Integrated Circuit (IC) bezeichnet [Alb07]. Der Chip lässt sich grob in drei Schichten teilen, dem Träger, der Schicht

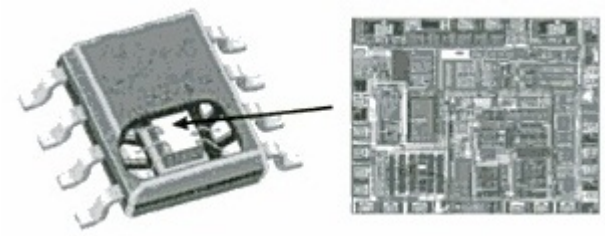

Abb. 3.1: Siliziumplättchen (Chip)

der Zellen und der Verdrahtungschicht. Der Träger, auch Substrat genannt, wird zu $99,9\%$ aus Silizium hergestellt. Auf ihm werden Transistoren integriert, die in logischen Einheiten, den Zellen, zusammengefasst werden. Anschließend werden mehreren Metallschichten zum Verdrahten aufgetragen [Alb07]. Da Chips immer kleiner, leistungsfähiger und billiger zu produzieren sein müssen, ist die Halbleiterfertigung einer der sich am schnellsten entwickelnden Industriezweige geworden. Die Entwicklung neuer Chips ist dabei eine ihrer Hauptaufgaben. Die Entwicklungsphasen eines neuen Chips sind in Abb. 3.2 schematisch dargestellt. Hierbei wird die integrierte Schaltung zuerst am Computer gestaltet (designed), dann werden im Frontend die ersten Chips

Abb. 3.2: Phasen der Chipentwicklung

gefertigt. Anschließend bekommen die Chips im Assembly ihr Kunststoffgehäuse und im Backend werden sie auf ihre Funktionsfähigkeit getestet [Alb07].

3.2 Modellierung elektronischer Schaltkreise

Die zu untersuchenden Schaltkreise bestehen aus sehr vielen Zellen und Leitungen, welche die Zellen miteinander verbinden. Jede Zelle besteht je nach Bauart und Zweck aus verschiedenen Transistoren und besitzt eine Anzahl an Anschlüssen. Diese werden auch Pins genannt und empfangen Signale oder leiten sie weiter. Sie werden in In- und Outpins unterschieden und für die Leitungen ergibt sich damit eine Richtung. In

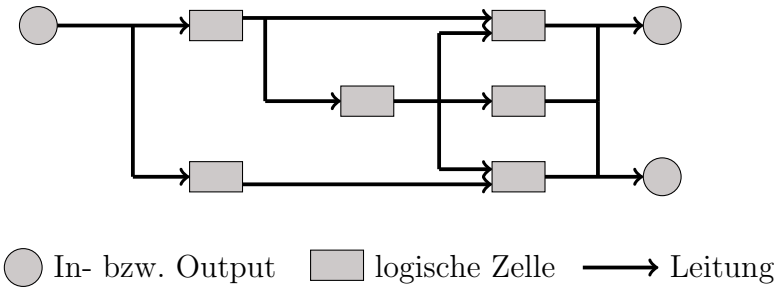

Abb. 3.3: Elektrische Schaltung

der Regel werden elektrische Schaltkreise mit Hypergraphen modelliert [Ber89]. Da die in der vorliegenden Arbeit verwendeten Verfahren jedoch an großen Schaltkreisen mit mehr als 10000 Zellen getestet werden, wurden die Schaltkreise als schlichte Graphen modelliert [2.1]. Dadurch verringern sich die Laufzeiten der angewendeten Algorithmen. Um die Schaltkreise adäquat als schlichte Graphen modellieren zu können, werden die zwei folgenden Forderungen aufgestellt:

- die Leitungen sind ungerichtet,
- Leitungen, die mehr als zwei Zellen miteinander verbinden werden in entsprechend viele Leitungen aufgeteilt, die genau zwei Zellen miteinander verbinden.

Damit ergibt sich für den Schaltkreis aus Abb. 3.3. folgende Darstellung als Graph:

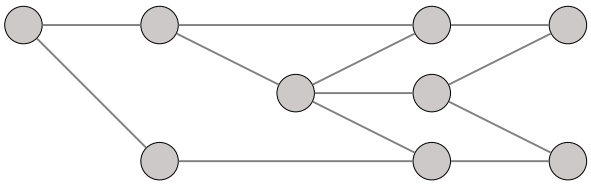

Abb. 3.4: Schlichter Graph

3.3 Aufgabenstellung

Die vorliegende Diplomarbeit setzt ihren Schwerpunkt auf die Untersuchung der zweiten Phase der Chipentwicklung, d.h. der Untersuchung des Layouts eines Chips. Das IC-Layout ist die Darstellung einer integrierten Schaltung mit Hilfe planarer geometrischer Formen, welche die zugehörigen elektronischen Bestandteile des integrierten Schaltkreises repräsentieren. Das Layout wird mit Hilfe der Bedingungen aus dem letzten Abschnitt als Graph modelliert. Die Adjazenzmatrix dieses Graphen ist relativ groß und dünn besetzt. Um innerhalb dieses Graphen ähnliche Grundstrukturen zu finden, muss er zunächst so geteilt werden, dass einzelne Komponenten miteinander verglichen werden können. Für das Zerlegen von Graphen stehen zurzeit eine große Auswahl an Algorithmen zur Verfügung [Sch07]. Da jedoch weder bekannt ist, in wie viele Teile sich der Layoutgraph G zerlegt, noch wie groß diese Teile sind, werden spektrale Verfahren zur Partitionierung verwendet. die vor allem für dünnbesetzte Matrizen geeignet sind. Dadurch kann die sogenannte *Eigengap-Heuristik* angewendet werden, welche die Anzahl der Komponenten schätzen soll [4.1].
Nachdem G in k Teilgraphen zerlegt ist, werden diese Teilgraphen mit einem Graphenähnlichkeitsmaß verglichen, einer suboptimalen *Graph Edit-Distanz* (GED). Um die suboptimale GED nutzen zu können, müssen für jeden der k Teilgraphen bestimmte Netzwerkzentralitäten berechnet bzw. eine Einbettung in den $\mathbb{R}^2$ bestimmt werden [4.2]. Dies geschieht über die jeweiligen Berechnungsalgorithmen der Zentralitäten bzw. einem Einbettungsalgorithmus von Kamada und Kawai [4.3.5]. Die GED gibt dabei nicht nur ein globales Ähnlichkeitsmaß zwischen den Graphen wieder, sondern gleichzeitig auch einen nachvollziehbaren Umformungsweg, der direkt einzelnen Komponenten des Graphen zugeordnet werden kann.

Da die suboptimale GED nicht symmetrisch und damit keine Metrik ist, ergeben sich insgesamt $k(k+1)$ Vergleiche. Aus den berechneten Distanzen wird eine Ähnlichkeitsmatrix S erstellt. Die Matrix S dient dabei einer Vorauswahl für die Erzeugung der Prototypen. Sie wird nun wieder mit Hilfe spektraler Verfahren zerlegt. Nach der Zerlegung der Matrix soll nun der dem Clusterzentrum am nächsten liegende Teilgraph in den Prototypen des Clusters umgewandelt werden. Dies geschieht mit Hilfe des Umformungsweges zwischen den Teilgraphen des Clusters, den *Graph Edit-Path* (GEP). Werden innerhalb der GEP eines Clusters Knoten vermehrt gelöscht bzw. eingefügt, werden diese auch im endgültigen Clusterprototypen gelöscht bzw. eingefügt.
Auch mit dem richtigen Ähnlichkeitsmaß ergibt sich das Problem, dieses Maß auf einen Graphen G anzuwenden, um die Ähnlichkeit einzelner Teilgraphen zu einander und zu G zu berechnen. Die Teilgraphen sollen nun gewissen Kriterien entsprechen, um ein aussagekräftiges Maß zu erhalten. Hierbei sollten bei der Suche nach Prototypen folgenden Voraussetzungen erfüllt sein:

- Die Teilgraphen sollen eine angemessene Größe bezüglich ihrer Knoten- und Kantenmengen haben. Ohne diese Forderung würden sich viele Graphen durch Teilgraphen mit zwei oder drei Knoten darstellen lassen bzw. lassen sich zu große Teilgraphen aufgrund hoher Rechenzeiten nicht miteinander vergleichen.

- Die Teilgraphen sollen möglichst viele Kanten des Originalgraphen enthalten. Dies soll analog zur obigen Bedingung dazu beitragen, den Originalgraphen möglichst genau abzubilden.

- Die Knotenmenge der Teilgraphen sollen paarweise disjunkt sein, d.h. enthält die Knotenmenge $V(T_1)$ des Teilgraphen T_1 den Knoten v, so enthält keine andere Knotenmenge diesen Knoten. Dies soll ermöglichen, den Graphen mit einer Menge von Teilgraphen aus verschiedenen Partitionierungsschritten abzubilden, ohne dass sich die Teilgraphen überlappen.

- Die Teilgraphen selbst sollen zusammenhängend sein.

Die Prototypen werden mit Hilfe des dem Clusterzentrum am nächsten liegenden Teilgraphen T und der Umformungswege der Teilgraphen des Cluster erstellt. Grundlage für den Prototypen ist dabei die Knoten- und Kantenmenge von T. Anschließend

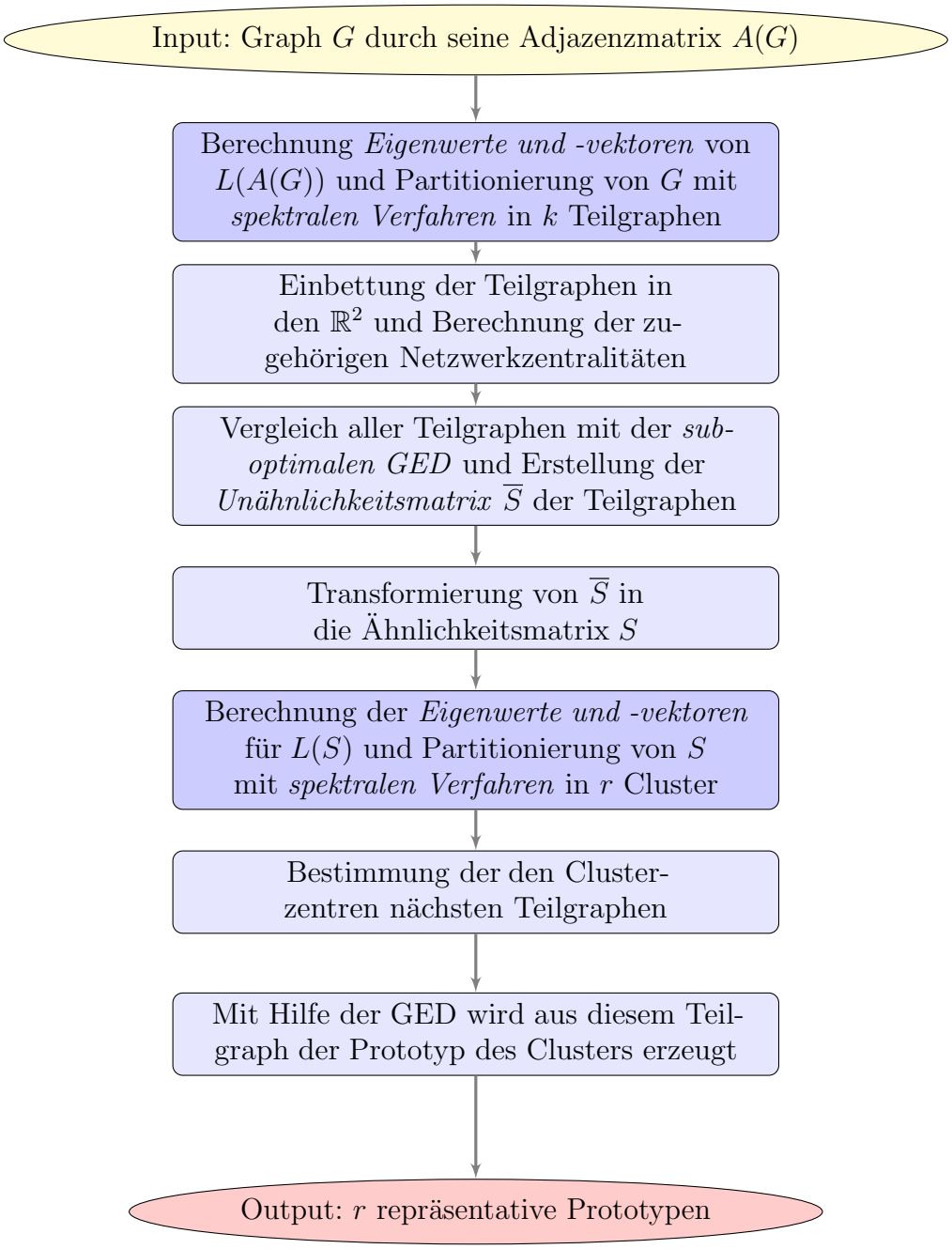

Abb. 3.5: Verfahren zur Identifizierung ähnlicher Grundstrukturen

werden jedem Knoten von T für jeden Transformationsschritt des Knotens, Kosten zugeteilt. Überschreitet ein Knoten aus T dabei eine vorgegebene Kostengrenze, wird er nicht mit in die Knotenmenge des Prototypen übernommen.

Die r repräsentativen Prototypen sollten den Graphen G gut abdecken, also eine größtmögliche Ähnlichkeit mit allen Teilgraphen von G haben. *Gute* Prototypen sollten also innerhalb eines Clusters ein Minimum an Graphoperationen bzw. Graph-

Edit-Distanzen haben.

Folgend soll der Prototyp des Clusters C der Größe 5 erzeugt werden. Die enthaltenen Teilgraphen sowie der erzeugte Prototyp sind in Abbildung 3.6 dargestellt. Tabelle 3.1 enthält die zugehörige Ähnlichkeitsmatrix des Clusters samt Prototyp.

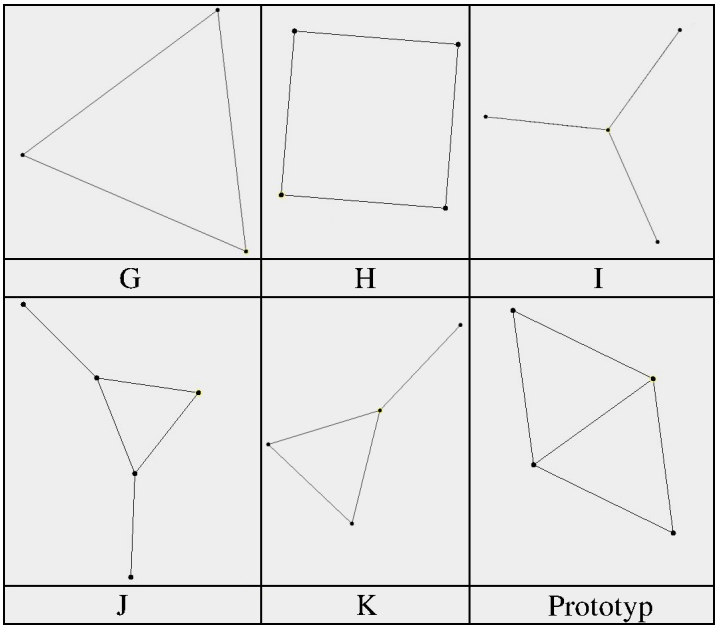

Abb. 3.6: Cluster C, bestehend aus den Teilgraphen G, H, I, J und K mit zugehörigem Prototyp

	G	H	I	J	K	Prototyp
G	1.0000	0.8646	0.8542	0.8625	0.9062	0.8875
H	0.8646	1.0000	0.8073	0.9000	0.7812	0.9500
I	0.8542	0.8073	1.0000	0.9125	0.9688	0.9500
J	0.8625	0.9000	0.9125	1.0000	0.9375	0.9125
K	0.9062	0.7812	0.9688	0.9375	1.0000	0.9750
Prototyp	0.8875	0.9500	0.9500	0.9125	0.9750	1.0000

Tab. 3.1: Ähnlichkeitsmatrix des Clusters C, samt Prototyp

Der Teilgraph K ist hierbei der zentrumsnächste Graph und der Prototyp besitzt die größte Summe der Ähnlichkeiten innerhalb des Clusters. Weiterhin könnte der Teilgraph H aus dem Cluster entfernt werden, da er die geringste Summe der Ähnlichkeiten innerhalb des Clusters hat, also am weitesten vom Zentrum entfernt ist.

4 Realisierung

In diesem Kapitel sollen die Verfahren vorgestellt werden, die für die Entwicklung eines Verfahrens zur Identifizierung ähnlicher Grundstrukturen genutzt werden. Dies sind die spektralen Clusterverfahren sowie ausgewählte Netzwerkzentralitäten und Ähnlichkeitsmaße.

4.1 Partitionieren

Clustern ist ein Synonym für das Zerlegen bzw. Partitionieren einer Menge von Daten in k *natürliche Gruppen*. Die Clusteranalyse teilt sich dabei in das *Finden* der Zerlegungen und der anschließenden Bewertung bezüglich der Güte der Partitionierung. Die folgenden Definitionen sind an [BE05, Seite 179] angelehnt.
Die in Forschung und Literatur vorgestellten Clusterverfahren lassen sich durch ihre verwendete Definition von Ähnlichkeit, ihrem Clustermodell, ihrem algorithmischen Vorgehen, d.h. ihrer sogenannten *Paradigmen* und ihrer Komplexität unterscheiden. Sie lassen sich in graphentheoretische, hierarchische, partitionierende und optimierende Verfahren sowie in weitere Unterverfahren einteilen. Partitionierende Verfahren verwenden eine gegebene feste Anzahl an Clustern und ordnen deren Elemente durch Umtauschfunktionen solange um, bis eine verwendete Zielfunktion ein Optimum erreicht. Bekanntester Vertreter dieser Verfahren ist der k-Means-Cluster-Algorithmus. Hierarchische Verfahren gehen von den trivialen Zerlegungen aus, d.h. der gröbsten Partition mit $k = 1$ (top-down) bzw. feinsten Partition mit $k = n$ (bottom-up). Durch Aufteilen bzw. Zusammenfassen lassen sich anschließend Cluster bilden. Einmal gebildete Gruppen können nicht mehr aufgelöst oder einzelne Elemente wie bei den partitionierenden Verfahren getauscht werden. In der vorliegenden Arbeit wird ein partitionierendes Verfahren genutzt, dass der *spektralen Partitionierung* [BE05]. Der in dieser Arbeit zu partitionierende Datensatz entspricht der Adjazenzmatrix des

Graphen G. Da dieser Graph relativ groß ist ($n >> 10000$), wurde ein geeignetes Verfahren gesucht, dass große Graphen in angemessener Zeit zerlegt. Das Verfahren der spektralen Partitionierung, welches von Fiedler [Fie73], [PSL90], [SM00] und Luxburg [Lux07] entwickelt wurde, zeichnet sich dabei durch einen geringen Rechenaufwand aus. Dabei nutzt das Verfahren die Eigenschaften der Laplacematrix des Graphen. Ziel ist es, eine natürliche Zerlegung des Graphen mit der spektralen Multisektion unter Einbezug der Eigengap-Heuristik zu erreichen. Dabei soll der Ursprungsgraph G iterativ partitioniert werden, bis jeder Teilgraph eine vorgegebene Größe erreicht hat. Bei der Betrachtung der daraus resultierenden Blöcke zeigen sich jedoch ungewollte Nebeneffekte [5.1], so dass anschließend der Graph mit der spektralen Bisektion iterativ in vergleichbar große Blöcke zerlegt wird, bis eine bestimmte Anzahl an Partitionen bzw. eine vorgegebene Partitionsgröße erreicht wurde. Da dieser Ansatz jedoch ähnliche Ergebnisse wie der vorherige liefert, wird der Graph mit einer Kombination beider Verfahren zerlegt. Dabei wird die spektrale Multisektion ohne Einbeziehung der Eigengap-Heuristik sondern mit festem $k = 2$ genutzt. Der Nachteil dieses Verfahrens ist jedoch, dass auf die Größe bzw. die Anzahl der entstehenden Partitionen kein direkter Einfluss genommen werden kann.
Die benötigten Eigenwerte und Eigenvektoren der Laplacematrizen können mit Hilfe der Potenzmethode bzw. des Arnoldiverfahrens berechnet werden [Saa03].
Die folgenden Abschnitte sollen die Herleitung und Algorithmen der spektralen Bisektion und Multisektion präsentieren. Die Herleitung der spektralen Bisektion ist [New10, Seite 364] entnommen.

4.1.1 Spektrale Bisektion

Um einen großen Graphen in angemessener Zeit zerlegen zu können, fällt die Wahl für das Partitionieren des Graphen auf das Verfahren der spektralen Bisektion. Das Verfahren, eingeführt von Fiedler [Fie73], nutzt die Eigenschaften der Laplacematrix eines Graphen. Im folgenden Abschnitt werden die spektrale Partitionierungsverfahren als Näherungslösungen für die Suche nach einem minimalen Schnitt vorgestellt. Soll der Graph in mehrere bzw. kleinere Teilgraphen zerlegt werden, wird die Bisektion hierarchisch angewandt. Dabei wird der Graph solange zerlegt, bis die gewünschte Größe bzw. Anzahl der Teilgraphen erreicht wird.
Betrachtet wird ein Graph G mit n Knoten und m Kanten und einer Partitionierung

$P(G)$ des Graphen in zwei Blöcke bzw. Cluster C und $\overline{C} = V \backslash C$. Die Größe des Schnittes $|(C, \overline{C})|$ der Partitionierung ist die Anzahl der Kanten zwischen C und $\overline{C}$, wobei für zwei Cluster gilt:

$$|(C, \overline{C})| = R = \frac{1}{2} \sum_{\substack{u \in C \\ v \in \overline{C}}} A_{u,v}, \tag{4.1}$$

hierbei normiert der Faktor $\frac{1}{2}$ die Summe, da sie jede Kante doppelt zählt.
Der Indikatorvektor $\vec{s}_u$ zeigt an, in welchem Cluster sich der Knoten u befindet. Es sei:

$$s_u = \begin{cases} 1, & u \in C, \\ -1, & u \in \overline{C}. \end{cases}$$

Weiterhin sei

$$\frac{1}{2}(1 - s_u s_v) = \begin{cases} 1, & u \text{ und } v \text{ in verschiedenen Clustern}, \\ 0, & u \text{ und } v \text{ im gleichen Cluster}. \end{cases}$$

Damit kann Gleichung (4.1) nun über alle u, v definiert werden:

$$R = \frac{1}{4} \sum_{u,v} A_{u,v}(1 - s_u s_v). \tag{4.2}$$

Der erste Teil der Summe lässt sich mit Hilfe des Kroneckerdeltas $\delta_{u,v}$ wie folgt umformulieren

$$\sum_{u,v} A_{u,v} = \sum_{u} \deg(u) = \sum_{u} \deg(u) s_u^2 = \sum_{u,v} \deg(u) \delta_{u,v} s_u s_v.$$

In Gleichung (4.2) eingesetzt ergibt sich daraus

$$\begin{aligned} R &= \frac{1}{4} \left(\sum_{u,v} \deg(u) \delta_{u,v} s_u s_v - \sum_{u,v} A_{u,v} s_u s_v \right) \\ &= \frac{1}{4} \sum_{u,v} (\deg(u) \delta_{u,v} s_u s_v - \sum_{u,v} A_{u,v}) s_u s_v \\ &= \frac{1}{4} \sum_{u,v} l_{u,v} s_u s_v = \frac{1}{4} \vec{s^T} L \vec{s}, \end{aligned} \tag{4.3}$$

mit L der Laplacematrix. Somit kann die Graphenbisektion in ein diskretes Optimierungsproblem umformuliert werden. Dabei wird der Faktor $\frac{1}{4}$ nicht weiter berücksichtigt, da er das Optimum nicht beeinträchtigt. Es ergibt sich die folgende Zielfunktion

$$\boxed{\min_{C_1 \subset V} \frac{1}{4} \vec{s^T} L \vec{s}.}$$

Da dieses Problem NP-schwer ist [New10], wird eine Näherungslösung gesucht, die leichter zu berechnen ist, jedoch die echte Lösung bestmöglich approximiert. Dazu wird das Optimierungsproblem relaxiert.

Eine Bedingung der Einträge von $\vec{s}$ ist, dass diese nur die Werte ± 1 annehmen dürfen. Betrachtet man $\vec{s}$ als einen Vektor im euklidischen Raum, folgt aus dieser Bedingung, dass der Vektor stets auf eine der 2^n Ecken eines n-dimensionalen Hyperwürfels um den Ursprung zeigt und immer die Länge $\sqrt{n}$ besitzt. Die Bedingung bezüglich der Richtung des Vektors wird nun relaxiert, so dass der Vektor in jede Richtung des n-dimensionalen Raumes zeigen kann. Die Länge des Vektors bleibt unverändert, da die Minimierung von R bei beliebiger Länge die triviale Lösung $\vec{s} = \vec{0}$ hätte. Die Einträge von $\vec{s}$ können nun jeden beliebigen Wert annehmen, solange gilt $|\vec{s}| = \sqrt{n}$ bzw. äquivalent dazu

$$\sum_i s_i^2 = n.$$

Der Vektor $\vec{s}$ zeigt nun nicht nur auf die Ecken des Hyperwürfels im n-dimensionalen Raum, sondern auf jeden Punkt der Oberfläche einer Hypersphäre um den Ursprung mit Radius $\sqrt{n}$. Die zweite Forderung an die Einträge von $\vec{s}$ ist, dass die Anzahl der

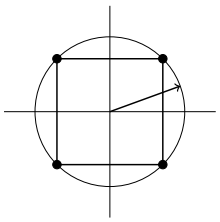

Abb. 4.1: Die Hypersphäre um den Hyperwürfel im n-dimensionalen Raum.

Einträge gleich -1 bzw. $+1$ genauso groß sein müssen, wie die gewünschten Größen der Cluster. Mit den beiden Clustergrößen n_1 und n_2 kann diese Bedingung wie folgt formuliert werden:

$$\sum_i s_i = n_1 - n_2,$$

oder in Vektorschreibweise als

$$\vec{1}^T \vec{s} = n_1 - n_2, \tag{4.4}$$

mit dem Einsvektor $\vec{1} = (1, 1, ..., 1)$. Diese zweite Bedingung wird nicht relaxiert und somit hat das relaxierte Optimierungsproblem folgende Form

$$\begin{aligned} \text{Zielfunktion:} \quad & \min_{C_1 \subset V} \tfrac{1}{4} \vec{s^T} L \vec{s} \\ \text{Nebenbedingung 1:} \quad & \sum_i s_i^2 = n, \\ \text{Nebenbedingung 2:} \quad & \sum_i s_i = n_1 - n_2. \end{aligned}$$

Nun wird die Lagrange-Relaxation mit den zwei Lagrangemultiplikatoren 2μ und λ angewendet. Statt der Lösung eines Optimierungsproblem mit der Zielfunktion $f(\vec{s}) = \vec{s^T} L \vec{s}$ und den zwei Nebenbedingungen $g_1(\vec{s}) = n = \sum_i s_i^2$ und $g_2(\vec{s}) = n_1 - n_2 = \sum_i s_i$ wird nun ein lokales Extremum der Lagrangefunktion

$$\begin{aligned} \Lambda(\vec{s}, \mu, \lambda) &= f(\vec{s}) - \lambda(g_1(\vec{s}) - n) - 2\mu(g_2(\vec{s}) - n1 - n_2) \\ &= \left[\sum_{j,k} L_{j,k} s_j s_k + \lambda \left(n - \sum_j s_j^2 \right) + 2\mu \left((n_1 - n_2) - \sum_j s_j \right) \right], \end{aligned}$$

gesucht. Dieses Minimum kann über den Gradienten der Lagrangefunktion berechnet werden:

$$\Lambda'(\vec{s}, \mu, \lambda) = 0. \tag{4.5}$$

Aus dieser Bedingung ergeben sich die folgenden drei Gleichungen:

$$\frac{\partial \Lambda}{\partial \vec{s}} = 2 \sum_j L_{i,j} s_j - 2\lambda s_i - 2\mu = 0, \tag{4.6}$$

$$\frac{\partial \Lambda}{\partial \lambda} = n - \sum_i s_i^2 = 0,$$

$$\frac{\partial \Lambda}{\partial \mu} = n_1 - n_2 - \sum_i s_i = 0.$$

Aus (4.6) folgt

$$\sum_j L_{i,j} s_j = \lambda s_i + \mu, \tag{4.7}$$

oder in Matrixschreibweise

$$L\vec{s} = \lambda\vec{s} + \mu\vec{1}.$$

Der Einsvektor $\vec{1}$ ist ein Eigenvektor der Laplacematrix mit Eigenwert 0, d.h. es gilt $L \cdot \vec{1} = 0$. Wird in (4.5) von links mit $\vec{1}^T$ multipliziert und entsprechend aus (4.4) substituiert, ergibt sich nun

$$\mu = -\frac{n_1 - n_2}{n}\lambda. \tag{4.8}$$

Wird der Vektor $\vec{x}$ wie folgt definiert

$$\vec{x} = \vec{s} + \frac{\mu}{\lambda}\vec{1} = \vec{s} - \frac{n_1 - n_2}{n}\vec{1}, \tag{4.9}$$

folgt aus (4.7) mit $L \cdot \vec{1} = 0$

$$L\vec{x} = L\left(\vec{s} + \frac{\mu}{\lambda}\vec{1}\right) = L\vec{s} = \lambda\vec{s} + \mu\vec{1} = \lambda\vec{x}.$$

Der Vektor $\vec{x}$ ist somit ein Eigenvektor der Laplacematrix mit Eigenwert λ, d.h. jeder Eigenvektor der Laplacematrix erfüllt die Gleichung (4.9).
Aus der Gleichung

$$\vec{1}^T\vec{x} = \vec{1}^T\vec{s} - \frac{\mu}{\lambda}\vec{1}^T\vec{1} = (n_1 - n_2) - \frac{n_1 - n_2}{n}n = 0. \tag{4.10}$$

folgt jedoch, das $\vec{x}$ orthogonal zu $\vec{1}$ ist. Der Eigenvektor $\vec{1}$ zum Eigenwert 0 entfällt damit. Die Wahl eines geeigneten Vektors folgt aus

$$R = \frac{1}{4}\vec{s}^T L\vec{s} = \frac{1}{4}\vec{x}^T L\vec{x} = \frac{1}{4}\lambda\vec{s}^T\vec{s}, \tag{4.11}$$

mit der Gleichung (4.8) folgt

$$\begin{aligned}\vec{x}^T\vec{x} &= \vec{s}^T\vec{s} + \frac{\mu}{\lambda}(\vec{s}^T\vec{1} + \vec{1}^T\vec{s}) + \frac{\mu^2}{\lambda^2}\vec{1}^T\vec{1} \\ &= n - 2\frac{n_1 - n_2}{n}(n_1 - n_2) + \frac{(n_1 - n_2)^2}{n^2}n \\ &= 4\frac{n_1 n_2}{n},\end{aligned}$$

und eingesetzt in (4.3)

$$R = \frac{n_1 n_2}{n}\lambda.$$

Die Schnittgröße ist damit proportional zum Eigenwert λ. Da die Schnittgröße R minimiert werden soll, wird der Eigenvektor zum kleinsten erlaubten Eigenwert der Laplacematrix gewählt. Da für die Eigenwerte von L gilt

$$0 \leq \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n,$$

und λ_1 entfällt, da $\vec{x}$ orthogonal zu diesem sein muss, ergibt sich ein minimaler Schnitt, wenn $\vec{x}$ proportional zum Eigenvektor mit dem Eigenwert λ_2 gewählt wird. Die zu $\vec{s}$ optimalen Näherungswerte ergeben sich damit aus Gleichung (4.8) wie folgt

$$\vec{s} = \vec{x} + \frac{n_1 - n_2}{n}\vec{1},$$

oder elementweise

$$s_i = x_i + \frac{n_1 - n_2}{n}.$$

Mit Hilfe der spektralen Bisektion kann eine optimale Näherungslösung gefunden werden, welche den Schnitt (4.1) minimiert. Dadurch kann der gegebene Graph in zwei Teilgraphen gegebener Größe zerlegt werden. Da in der vorliegenden Arbeit ähnliche Teilgraphen gesucht werden, d.h. Teilgraphen, die auch eine ähnliche Größe besitzen sollten, wird der Graph mit Hilfe der Bisektion hierarchisch in vergleichbar große Blöcke zerlegt. Hierfür wird der **Median** (Zentralwert) $\tilde{x}$ genutzt. Dafür werden die Einträge des Eigenvektors $x \in \mathbb{R}^n$ aufsteigend angeordnet und der Median $\tilde{x}$ ergibt sich dann als

$$\tilde{x} = \begin{cases} x_{\frac{n+1}{2}}, & n \text{ ist ungerade,} \\ x_{\frac{n}{2}+1}, & \text{sonst.} \end{cases}$$

Da für die spektrale Bisektion nur der zweite Eigenvektor der Laplacematrix eines Graphen berechnet werden muss und anschließend der Median des Eigenvektors ermittelt wird, ist das Verfahren sehr schnell und damit geeignet, um große Graphen in angemessener Zeit zu partitionieren.

Algorithmus 4.1.1: Algorithmus für die spektrale Bisektion

Eingabe: Adjazenzmatrix $A \in \mathbb{R}^{n \times n}$, Anzahl k der zu konstruierenden Cluster

Ausgabe: Zwei Cluster C_1, C_2, mit den Knoten (Objekten) aus A

Bestimmung der Laplace-Matrix L;

Berechnung des zweiten Eigenvektors u der Laplacematrix;

Bestimmung des Median $\tilde{x}$ von u;

for $i = 1, ..., n$ **do**

  if $x(i) \leq \tilde{x}$ **then**

    Das Objekt i wird dem Cluster C_1 zugeordnet;

  else

    Das Objekt i wird dem Cluster C_2 zugeordnet;

  end

end

4.1.2 Spektrale Multisektion

Die Herleitung der spektralen Multisektion auf k Cluster des Graphen G erfolgt analog zur Herleitung der spektralen Bisektion von Newman. Eine weitere Herleitung wird in [Lux07] vorgestellt. Für die Größe des Schnittes des Graphen in k Cluster gilt nun nach (4.1)

$$|(C_1, C_2, ..., C_k)| = R = \frac{1}{2} \sum_{j=1}^{k} |(C_j, \overline{C_j})|,$$

mit $\overline{C_j} = V \backslash C_j$ Die Indikatormatrix $S = (s_{i,j})$ zeigt nun jedoch an, in welchem der k Cluster C_j sich der Knoten i befindet.

$$s_{i,j} = \begin{cases} 1, & i \in C_j, \\ 0, & i \in \overline{C_j}. \end{cases}$$

Aus der spektralen Bisektion ist bekannt:

$$|(C_j, \overline{C_j})| = \frac{1}{4} \vec{h_j}^T L \vec{h_j}, \tag{4.12}$$

mit dem Vektor $\vec{h_j}$

$$h_j = \begin{cases} 1, & i \in C_j \\ -1, & i \in \overline{C_j}. \end{cases}$$

Zusätzlich werden die Vektoren $\vec{s_j}$ und $\vec{p_j}$ wie folgt definiert

$$s_{i,j} = \begin{cases} 1, & i \in C_j \\ 0, & i \in \overline{C}, \end{cases}$$

Damit ergibt sich

$$\vec{h_j} = 2\vec{s_j} - \vec{1}.$$

Aus (4.12) und $L \cdot \vec{1} = 0$ folgt dann

$$\begin{aligned} |(C_j, \overline{C_j})| &= \frac{1}{4}\vec{h_j}^T L \vec{h_j} \\ &= \frac{1}{4}(2\vec{s_j} - \vec{1})^T L (2\vec{s_j} - \vec{1}) \\ &= \vec{s_j}^T L \vec{s_j}. \end{aligned}$$

Damit folgt für die Größe des Schnittes:

$$R = \sum_{j=1}^{k} R_j = \frac{1}{2} \sum_{j=1}^{k} \vec{s_j}^T L \vec{s_j}.$$

Statt die Summe als Ganzes zu minimieren, werden nun die Summanden minimiert, da die Summe der minimalen Summanden minimal ist. Analog zur Bisektion wird die folgende Nebenbedingung für jeden der k Summanden vereinbart:

$$\sum_{i=1}^{n} s_{i,j} = n_j,$$

mit $\sum_{j=1}^{k} n_j = n$. Der Vektor $\vec{x_j}$ wird nun definiert als:

$$\vec{x} = \vec{s} + \frac{\mu}{\lambda}\vec{1} = \vec{s} - \frac{2n_j - n}{n}\vec{1}.$$

Aus der Lagrange-Relaxation mit anschließender Ableitung ergibt sich nun

$$L\vec{x_j} = L\left(\vec{s_j} + \frac{\mu_j}{\lambda_j}\vec{1}\right) = L\vec{s_j} = \lambda_j \vec{s_j} + \mu_j \vec{1} = \lambda_j \vec{x_j},$$

Der Vektor $\vec{x_j}$ ist somit ein Eigenvektor der Laplacematrix mit Eigenwert λ_j. Da aus (4.10) folgt das $\vec{x_j}$ orthogonal zum Einsvektor $\vec{1}$ ist, entfällt der Eigenvektor $\vec{1}$ zum Eigenwert 0. Die Wahl des geeigneten Vektors folgt aus (4.11) und

$$\begin{aligned}\vec{x_j}^T\vec{x_j} &= \vec{s_j}^T\vec{s_j} + \frac{\mu_j}{\lambda_j}(\vec{s_j}^T\vec{1} + \vec{1}^T\vec{s_j}) + \frac{\mu_j^2}{\lambda_j^2}\vec{1}^T\vec{1} \\ &= n_j - \frac{n_j}{n}(n_j + n_j) + \frac{(n_j)^2}{n^2}n \\ &= \frac{n_j(n-1)}{n},\end{aligned}$$

und eingesetzt in (4.3)

$$R_j = \frac{n_j(n-1)}{n}\lambda.$$

Die Schnittgröße ist wieder proportional zum Eigenwert λ. Da die Summe der Schnittgrößen R minimiert werden soll, werden Vektoren gewählt, welche proportional zu den Eigenvektoren mit den k kleinsten erlaubten Eigenwerten der Laplacematrix sind.

Die zu $\vec{s_j}$ optimalen Näherungswerte ergeben sich damit wie folgt

$$\vec{s_j} = \vec{x_j} + \frac{2n_j - n}{n}\vec{1}.$$

Damit ergibt sich der folgende Algorithmus für die spektrale Multisektion.

Algorithmus 4.1.2: Algorithmus für die spektrale Multisektion

Eingabe: Adjazenzmatrix $A \in \mathbb{R}^{n\times n}$, Anzahl k der zu konstruierenden Cluster

Ausgabe: Cluster $A_1, ..., A_k$ mit $A_i = \{j | y_j \in C_i\}$

Berechnung der Laplace-Matrix L;

Bestimmung die ersten k Eigenvektoren $u_1, ..., u_k$ von der Laplacematrix;

Erzeugung der Matrix $U \in \mathbb{R}^{n\times k}$, welche die Vektoren $u_1, ..., u_k$ als Zeilen enthält;

Für $i = 1, ..n$, sei $y_i \in \mathbb{R}^k \, \forall i = 1, ..., n$ der zur i-ten Spalte von U zugehörige Vektor;

Zuordnung der Punkte $(y_i)_{i=1,...,n}$ mittels k-Means-Algorithmus zu den Clustern $C_1, ..., C_k$;

Der k-Means-Cluster-Algorithmus

Der k-Means-Algorithmus ist ein Verfahren der Clusteranalyse, der aus einer Menge von ähnlichen Objekten eine vorher bekannte Anzahl von k Gruppen (Clustern) bildet. Der Algorithmus ist eine der am häufigsten verwendeten Techniken zur Gruppierung von Objekten, da er schnell zu implementieren ist und ebenso schnell die Zentren

(*Means*) der Cluster findet. Der k-Means-Algorithmus findet immer eine Lösung, welche nicht zwangsweise optimal sein muss. Eine optimale Partitionierung zu finden, gehört zur Klasse der NP-schweren Probleme [KMN+02], [BE05].
Die Lösungen des k-Means-Algorithmus hängen dabei stark von den Startpunkten der verwendeten Daten und der gewählten Anzahl der Cluster ab. So ergeben sich für verschiedene Startpositionen unterschiedliche Partitionierungen, da z.B. die Clusterzentren unterschiedlich berechnet wurden. Um die daraus resultierende Varianz zu minimieren, wird der Algorithmus mehrmals hintereinander ausgeführt. Ebenso kann die ungünstige Wahl der Anzahl der zu bestimmenden Cluster zu schlechten Ergebnissen führen. Um eine passende Wahl bezüglich der Clusteranzahl zu treffen, wird diese normalerweise schrittweise erhöht. Dadurch wird die totale Varianz der Daten schnell zurückgehen. Wenn der Rückgang stagniert, ist ein guter Wert für k erreicht. Im Rahmen der spektralen Partitionierung kann diese aufwendige Bestimmung von k mit Hilfe der Eigengap-Heuristik umgangen werden.
Der k-Means-Algorithmus benötigt eine Distanzfunktion und eine Funktion zur Berechnung der Cluster-Schwerpunkte. In der vorliegenden Arbeit wurde hierfür in beiden Fällen die euklidische Norm verwendet.

Algorithmus 4.1.3: k-Means-Cluster-Algorithmus

Eingabe: n Objekte $x \in \mathbb{R}^d$, Anzahl k der Cluster
Ausgabe: Partitionierung der n Objekte in k Cluster (Blöcke)
Die k Means $m_1, ..., m_k$ werden zu Beginn zufällig verteilt;
while *Objekte werden anderen Clustern zugeordnet* **do**
  for $j = 1 \to n$ **do**
    Zuordnung des Objekts x_j zu dem Cluster C, dessen Schwerpunkt am nächsten liegt. $C_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \right\| \forall\, i^* = 1, \ldots, k \right\}$;
  end
  for $i = 1 \to k$ **do**
    Neuberechnung der Schwerpunkte m_i der Cluster, so dass diese im Zentrum der Cluster liegen: $\mathbf{m}_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{\mathbf{x}_j \in C_i^{(t)}} \mathbf{x}_j$;
  end
end

Die Eigengap-Heuristik

Sind keine zusätzlichen Informationen über die dem Graphen zugrundeliegenden Strukturen bekannt, ist es oft schwierig, die richtige Anzahl k Cluster einer Partitionierung zu wählen, die den Graph zerlegen soll. Die *Eigengap-Heuristik* aus der

Spektraltheorie soll nun ein geeignetes k finden. Nach Lemma 2.2. gilt, dass die Laplacematrix die gleiche Anzahl an Eigenwerten $\lambda = 0$ wie Komponenten besitzt. Die Eigengap-Heuristik sucht nun nach einem k, für das gilt $\lambda_1, ..., \lambda_k$ sind klein und λ_{k+1} ist relativ groß. Es wird eine große Differenz zwischen zwei aufeinanderfolgenden Eigenwerten gesucht, der sogenannte *Eigengap*.
Für den Idealfall, dass der Graph aus k zusammenhängenden Komponenten besteht, wird ein eindeutiger Eigengap gefunden. Je stärker die Komponenten allerdings miteinander verbunden sind, desto schwieriger wird es, ein geeignetes k mit Hilfe des Eigengaps zu finden, da die Differenzen zwischen den Eigenwerten relativ kleiner werden. Mehr Einzelheiten bezüglich der Eigengap-Heuristik finden sich in [Zag05] und [MJ97].

4.2 Netzwerkzentralitäten

Netzwerkzentralitäten wurden in der Datenverarbeitung entwickelt, um genauere Aussagen über Netzwerke und ihre Strukturen treffen zu können. Im Mittelpunkt vieler dieser Zentralitäten stehen die Aspekte der Wichtigkeit eines Knotens innerhalb des Netzwerkes und die Vernetzung des Netzwerkes als Ganzes.
Da in den verschiedenen Forschungsgebieten z.T. sehr verschiedene Netzwerke existieren, z.B. das Web-Netzwerk im Vergleich zu Straßennetzwerken, wurden viele netzwerkspezifische Zentralitäten für spezzielle Netzwerke entwickelt. Daher muss bei der Auswahl einer geeigneten Netzwerkzentralität zum Vergleich zweier oder mehrerer Netzwerke neben ihren strukturellen Eigenschaften, auch deren Herkunft miteinbezogen werden.
Der folgende Abschnitt soll einen kurzen Einblick auf die von der *GED* genutzten Netzwerkzentralitäten geben. Die folgenden Netzwerkzentralitäten wurden zumeist verwendet, um den Webgraphen zu beschreiben und sind u.a. [BE05] und [New10] entnommen.

4.2.1 Knotengradzentralität

Eine einfache Zentralität ist die Knotengradzentralität. Hierbei erhält ein Knoten u eine zu $\deg(u)$ proportionale Zentralität. Analog erhält u in gerichteten Netzwerken eine zu $\deg_{in}(u)$ bzw. $\deg_{out}(u)$ proportionale Zentralität. Trotz ihrer Einfachheit

liefert diese Zentralität bereits gute Aussagen über die Wichtigkeit einzelner Knoten. So kann z.B. in sozialen Netzwerken davon ausgegangen werden, dass Mitglieder mit vielen Bekannten einflussreicher, d.h. zentraler sind, als andere.
Die Knotengradzentralität kann in $\mathcal{O}(n)$ berechnet werden.

4.2.2 Eigenvektorzentralität

Die Eigenvektorzentralität wurde als Erweiterung der Knotengradzentralität 1987 von Phillip Bonacich entwickelt [BE05], [New10]. Während die Knotengradzentralität dem Knoten u für jeden Nachbarn den gleichen *Zentralitätszuwachs* zuteilt, unterscheidet die Knotenzentralität zwischen wichtigen und weniger wichtigen Nachbarn. Der Knoten u erhält damit einen zum Knotengrad seines Nachbarn proportionalen *Zentralitätszuwachs*. Die Eigenvektorzentralität ist damit eine sogenannte Feedback-Zentralität.
Zuerst wird die Eigenvektorzentralität aller Knoten geschätzt, z.B. kann die Zentralität x_i aller Knoten i auf eins gesetzt werden. Diese Schätzung wird nun iterativ verbessert. Die neue Zentralität x'_i ergibt sich aus

$$x'_i = \sum_j a_{i,j} x_j,$$

mit den Einträgen $a_{i,j}$ der Adjazenzmatrix A. In Matrixschreibweise ergibt sich $\vec{x'} = A\vec{x}$, mit dem Vektor $\vec{x}$, der die Zentralitäten x_i enthält. Nach t-facher Wiederholung der Neuberechnung der Schätzung ergibt sich der Zentralitäsvektor $\vec{x}(t)$

$$\vec{x}(t) = A^t \vec{x}(0).$$

Der Vektor $\vec{x}(0)$ lässt sich nun wie folgt als Linearkombination der Eigenvektoren $\vec{v}_i$ der Adjazenzmatrix entsprechender Wahl der Konstanten c_i darstellen

$$\vec{x}(0) = \sum_i c_i \vec{v}_i.$$

Dann ist

$$\vec{x}(t) = A^t \sum_i c_i \vec{v}_i = \sum_i c_i \lambda_i^t \vec{v}_i = \lambda_n^t \sum_i c_i \left[\frac{\lambda_i}{\lambda_n}\right]^t \vec{v}_i,$$

mit den Eigenwerten λ_i von A, wobei λ_n der größte Eigenwert von A sei. Da $\lambda_i/\lambda_n < 1 \, \forall i \neq n$ verschwinden alle Summanden bis auf den n-ten für eine steigende Anzahl der Iterationen

$$\lim_{t \to \infty} \vec{x}(t) = c_n \lambda_n^t \vec{v}_n.$$

Der Zentralitätsvektor ist damit direkt proportional zum größten Eigenwert von A. Damit erfüllt die Eigenvektorzentralität $\vec{x}$ folgende Gleichung

$$A\vec{x} = \lambda_n \vec{x}.$$

Die Zentralität von x_i des Knotens i ist damit proportional zur Summe der Zentralitäten seiner Nachbarn

$$x_i = \lambda_n^{-1} \sum_j a_{i,j} x_j,$$

womit die Eigenvektorzentralität entsprechend groß ist, wenn ein Knoten viele unwichtige oder wenige wichtige Nachbarn hat, bzw. viele wichtige Nachbarn besitzt. Um die Eigenvektorzentralitäten eines Netzwerkes mit denen eines anderen besser miteinander vergleichen zu können, wird sie anschließend mit der größten Zentralität innerhalb des Graphen normiert, so dass gilt $x_i \in [0, 1]$. Der größte Eigenwert kann hierbei mit dem Arnoldi-Verfahren berechnet werden [Saa03]. Damit ergibt sich für die Eigenvektorzentralität eine Laufzeit von $\mathcal{O}(n^3 \cdot k)$, wobei k gleich der Anzahl der gegebenen Iterationen sein soll.

4.2.3 PageRank

Die Eigenvektorzentralität besitzt leider die Eigenschaft, dass ein Knoten mit hoher Zentralität und vielen Nachbarn diese hohe Zentralität direkt auf seine Nachbarn überträgt. Dies kann dazu führen, dass z.B. unwichtige Knoten mit nur einem wichtigen Nachbarn eine höhere Zentralität besitzen als Knoten mit mehr Nachbarn. Deren Gesamtzentralität wäre hierbei jedoch niedriger. Daher wird für gerichtete Graphen die Idee eingeführt, dass die von Nachbarn erhaltene Zentralität proportional zu den ausgehenden Kanten des Nachbarn sein soll. So geben Knoten mit hoher Zentralität nur einen kleinen Teil ihrer Zentralität an ihre Nachbarn ab. Diese Zentralität ist speziell für den Webgraphen entwickelt worden und wird PageRank genannt. PageRank wurde 1998 von Page et al. entwickelt [ANTT02], [JW02] und ist Teil des

Algorithmus der Suchmaschine Google [New10].
Die Zentralität berechnet sich damit wie folgt

$$x_i = \begin{cases} \alpha \sum_j a_{i,j} x_j + \beta, & k_j^{out} = 0, \\ \alpha \sum_j a_{i,j} \frac{x_j}{k_j^{out}} + \beta, & \text{sonst.} \end{cases}$$

Im Fall $k_j^{out} = 0$ folgt aus $a_{i,j} = 0 \, \forall i$, dass $x_i = 0$. In Matrixschreibweise ergibt sich

$$\vec{x} = \alpha A K^{-1} \vec{x} + \beta \vec{1},$$

mit der Diagonalmatrix K mit den Einträgen $K_{i,i} = \max(k_j^{out}, 1)$. Ist $\beta = 1$ folgt

$$\vec{x} = (I - \alpha A K^{-1}) \vec{1} = K(K - \alpha A)^{-1} \vec{1}.$$

Die Konstante $\alpha \geq 0$ sei hierbei ein Dämpfungsfaktor und $\beta \geq 0$ eine konstante *Zusatzzentralität*, die jeder Knoten *extra* erhält. Für $\beta = 0$ ergibt sich für ungerichtete Graphen, d.h. $x_i = k_i = k_i^{out}$, wieder eine der Knotengradzentralität ähnliche Zentralität. Damit ergibt sich für PageRank eine Laufzeit von $\mathcal{O}(n^2 \cdot t)$, wobei t gleich der Anzahl der gegebenen Iterationen sein soll.

4.2.4 Hub- & Authority-Gewichte

Kurz nach Veröffentlichung von PageRank wurde von Kleinberg 1999 [Kle99] die Idee der Bestimmung der Zentralität von Webknoten mit Hilfe zweier Kriterien eingeführt. Diese zwei Kriterien sind die Hub- und Authority-Gewichte eines Knotens. Dazu werden die Knoten eines Netzwerkes in zwei Gruppen aufgeteilt. Dies geschieht mit Hilfe von Informationen über den Inhalt der Knoten bzw. Seiten im Internet. Authorities sind hierbei Knoten, die wichtige Informationen enthalten, z.B. Webseiten über wissenschaftliche Arbeiten. Hubs sind jedoch Knoten, die auf viele Authorities zeigen, z.B. Suchmaschinen. Zwischen Hubs und Authorities gilt, dass *ein guter Hub auf viele gute Authorities zeigt* und *eine gute Authority von vielen guten Hubs angezeigt wird*. Diese Grundidee führt zu Kleinbergs Zentralitätsalgorithmus, dem *hyperlink-induced topic search*, kurz *HITS*.

Die Berechnung der Hub- bzw. Authority-Gewichte y_i bzw. x_i erfolgt iterativ über

$$x_i = \sum_j a_{i,j} y_j,$$
$$y_i = \sum_j a_{i,j} x_j,$$

wobei die Gewichte nach jedem Schritt normalisiert werden, so dass gilt $x_i \in [0,1]$ und $y_i \in [0,1]$. Ein Vorteil gegenüber der Eigenvektorzentralität liegt darin, dass Knoten, welche an stark vernetzte Komponenten angrenzen, jedoch nicht auf diese zeigen, dennoch keine geringe Zentralität besitzen müssen. Ein Knoten, der nicht von vielen anderen angezeigt wird, d.h. ein kleines Authoritygewicht hat, kann aber dennoch ein hohes Hub-Gewicht haben, wenn er viele Knoten anzeigt. Analoges gilt für den Fall, dass ein Knoten selbst auf wenige oder keinen Knoten zeigt, aber selbst von vielen Knoten angezeigt werden kann. Hier ergibt sich für ein Netzwerk N eine Laufzeit von $\mathcal{O}(n \cdot \tilde{\vartheta}(N)k)$, wobei k gleich der Anzahl der gegebenen Iterationen sei. Weiterhin ist $\tilde{\vartheta}(N) = \max\{\vartheta(N)_{in}, \vartheta(N)_{out}\}$ gleich dem Maximum der durchschnittlich ein- und ausgehenden Knotengrade.
Dieses Verfahren wurde u.a. in [BVD04] und [Zag05] betrachtet und weiterentwickelt. Die Gewichte werden u.a. nicht mehr nur für die Knoten, sondern zusätzlich für die Kanten mit berechnet.

4.2.5 Die betweenness-Zentralität

Eine andere Art der Zentralität ist die betweenness-Zentralität, bei der gemessen wird, wie oft ein Knoten Teil eines kürzesten Pfades zwischen zwei Knoten eines Netzwerkes liegt. Die Idee der *betweenness* stammt von Anthonisse [Ant71] und wurde erst unter Freeman im Jahr 1977 bekannt [Fre77].
Grundgedanke dieser Zentralität ist es, das ein Knoten wichtiger, d.h. zentraler ist, wenn er auf möglichst vielen kürzesten (geodäsischen) Wegen zwischen zwei Knoten liegt. Es sei z.B. ein Netzwerk gegeben, in welchem über die Bögen Nachrichten zwischen den Knoten weitergeleitet werden. Ein Knoten, der auf vielen kürzesten Wegen liegt, muss von vielen Nachrichten passiert werden. Kann dieser Knoten nun z.B. die Nachrichten lesen oder sonst irgendwie beeinflussen, erhält er eine gewisse Wichtigkeit aus seiner Position im Netzwerk. Zur Berechnung der *betweenness* sei

nun

$$n_{st}^{i} = \begin{cases} 1, & \text{der Knoten } i \text{ liegt auf dem kürzesten Pfad von } s \text{ nach } t, \\ 0, & \text{sonst.} \end{cases}$$

Liegen s und t in verschiedenen Komponenten, d.h. es existiert kein $st-$Pfad, ist n_{st}^{i} ebenfalls null. Die *betweenness*-Zentralität ergibt sich nun als

$$x_i = \sum_{st} n_{st}^{i}.$$

Für die Berechnung der betweenness-Zentralitäten eines ungewichteten bzw. gewichteten Netzwerkes ergibt sich in eine Laufzeit von $\mathcal{O}(n \cdot m)$ bzw. $\mathcal{O}(n \cdot m + n^2 \cdot \log(n))$ [Bra01]. Die Kanten eines gewichteten Netzwerkes werden zusätzlich durch eine Funktion $\omega : E \mapsto \mathbb{R}$ verschieden gewichtet.

4.3 Ähnlichkeit in Graphen

Während sich Knoten innerhalb eines oder mehrerer Graphen mit Hilfe der Netzwerkzentralitäten des vorherigen Abschnitts miteinander vergleichen lassen, ist der Vergleich von Graphen ungemein schwerer. Eine Übersicht bekannter Ähnlichkeitsmaße findet man in [Sch04], [Jol01] [WM03]. Folgender Abschnitt ist in Anlehnung an [BE05, Seite 332] verfasst.

Das Problem der Graphenisomorphie beschäftigt sich mit der Frage, ob zwei Graphen strukturell gleich sind. Eine natürliche Vereinfachung ist es zu untersuchen, wie sehr sich zwei Graphen ähnlich sind. Graphenähnlichkeit wird oft als Graphmatching beschrieben.

Ein Vorteil der Graphenähnlichkeit gegenüber der Graphenisomorphie ist die Fähigkeit, weniger anfällig für Messfehler bzw. weißen Rauschen innerhalb der zu betrachtenden Daten zu sein, die bei der Gewinnung reeller Daten auftauchen können. Isomorphe Graphen können durch solche Fehler nicht mehr erkannt werden. Ein Alternative für die Bestimmung der Graphenisomorphie ist die Einführung eines Graphähnlichkeitsmaß. Um ein aussagekräftiges Ähnlichkeitsmaß zu erhalten, sollte es bestimmte Bedingungen erfüllen.

Definition 4.1

Seien G_1, G_2 und G_3 Graphen. Eine Funktion $d : G_1 \times G_2 \to \mathbb{R}_0^+$ wird als **Graphähnlichkeitsmetrik** bezeichnet, wenn es folgende Bedingungen erfüllt:

$$\begin{aligned} \textit{Reflexivität:} \quad & d(G_1, G_2) = 0 \Leftrightarrow G_1 \cong G_2, \\ \textit{Symmetrie:} \quad & d(G_1, G_2) = d(G_2, G_1), \\ \textit{Dreiecksungleichung:} \quad & d(G_1, G_2) + d(G_2, G_3) \geq d(G_1, G_3). \end{aligned}$$

Aussagekräftige Graphenähnlichkeitsmaße sind im Allgemeinen schwer zu berechnen, da aus der Reflexivitätsbedingung eine Lösung für das Graphenisomorphieproblem folgen muss. Im folgenden Abschnitt sollen drei Ähnlichkeitsmaße vorgestellt werden. Das erste Maß beruht auf der Größe des größten gemeinsamen Teilgraphen, die zwei letzteren auf der sogenannten Edit-Distanz. Die ersten beiden Ähnlichkeitsmaße sind [BE05, Seite 332], das letzte aus [BKL08]entnommen.

4.3.1 Maximum Common Subgraph (MCS)

Definition 4.2

Ein Graph G wird **gemeinsamer Teilgraph** der beiden Graphen G_1 und G_2 genannt, wenn G Teilgraph von G_1 und G_2 ist. Ein gemeinsamer Teilgraph ist **maximal**, d.h. ein MCS, wenn kein anderer gemeinsamer Teilgraph von G_1 und G_2 mit mehr Knoten existiert.

Die MCS-Distanz zweier nichtleerer Graphen ergibt sich wie folgt

$$d_{MCS}(G_1, G_2) = 1 - \frac{|V(MCS(G_1, G_2)|)}{max\{V(G_1), V(G_2)\}}.$$

Dieses Ähnlichkeitsmaß ist eine Graphähnlichkeitsmetrik und nutzt viele strukturelle Eigenschaften der Graphen, da die Struktur des MCS in beiden Graphen vorkommt. Für die Berechnung der MCS-Distanz zwischen zwei Graphen ergibt sich eine exponentielle Laufzeit [Sch04, Seite 56], dass MCS-Problem ist damit NP-schwer und praktisch nur auf relativ kleine Graphen anwendbar [BS98],[MB98].

4.3.2 Maß von Padadopoulos und Manolopoulos

Eine allgemeine und flexible Methode um Graphen miteinander zu vergleichen, ist die der *Edit-Distanz*. Dazu sei eine Menge erlaubter Graphoperationen auf den Graphen definiert [2.1.3]. Die Distanz zwischen zwei Graphen ist nun die minimale Anzahl an Operationen, um einen Graphen auf den anderen abzubilden. Ursprünglich wurde die Edit-Distanz von Levenshtein entwickelt [4.4] und wird vor allem zum Vergleich von Zeichenketten verwendet [4.4].
Die hier vorgestellten Edit-Distanzen nutzen als Operationen das Löschen, Einfügen und Substituieren von Knoten und Kanten. Weitere sinnvolle Graphoperationen wären die *Kontraktion* und *Fusion* von Knoten [Tit03]. Den Operationen können beliebige Kosten zugeordnet werden, die jedoch im Verhältnis zur Wichtigkeit der einzelnen Transformation stehen sollten. Die Distanz berechnet sich dann als das Minimum der Summe der Kosten, um einen Graphen in den anderen zu transformieren und ist eine Graphähnlichkeitsmetrik.
Eine solche Edit-Distanz wurde von Padadopoulos und Manolopoulos 1999 vorgestellt [PM99]. Sie nutzt folgenden Knotenoperationen, welche die entsprechenden Kantenoperationen enthalten, z.B. Löschen aller inzidenten Kanten beim Löschen eines Knoten:

- Knoten löschen,
- Knoten einfügen,
- Knoten substituieren.

Zur Bestimmung der Ähnlichkeit werden die Histogramme zweier Graphen miteinander verglichen. Histogramme sind geordnete Vektoren, welche die Knotengrade der jeweiligen Graphen in abnehmender Reihenfolge enthalten. Enthält ein Graph weniger Knoten als der mit ihm zu vergleichende, wird das Histogramm mit Nullen aufgefüllt, bis beide Histogramme gleich lang sind. Die Ähnlichkeit ergibt sich als Betrag der Differenz der Manhattanmetrik beider Histogramme, d.h.

$$d_{PM}(G_1, G_2) = ||V(G_1)| - |V(G_2)||.$$

Dieses Maß transformiert somit nicht die Graphen direkt ineinander, sondern nur ihre Histogramme und damit die Anzahl der Operationen, um ein Histogramm in ein

anderes zu transformieren. Dadurch fließen relativ wenig strukturelle Informationen in die Berechnung der Ähnlichkeit. Jedoch lässt sich dieses Maß relativ schnell an bestimmte Klassen von Graphen anpassen, indem statt der Knotengrade andere Netzwerkeigenschaften (z.B. die Exzentrizitäten) miteinander verglichen werden.

4.3.3 Graph-Edit-Distanz

Die Grundidee von Graph Edit-Distanz, einer speziellen Edit-Distanz, ist es, die Unähnlichkeit zweier Graphen mit Hilfe der minimalen Anzahl an Umformungen, um einen Graphen in einen anderen zu transformieren, zu beschreiben. Die folgenden Ausführungen sind an [BKL08] angelehnt.
In der Regel werden als Umformungsoperationen das *Einfügen*, *Löschen* und *Substituieren* von Knoten und Kanten verwendet [siehe 2.1.3]. Hierbei wird folgende Notation für die Substitution zweier Knoten u und v mit $u \to v$, das Löschen eines Knoten u mit $u \to \epsilon$ und das Einfügen eines Knotens v mit $\epsilon \to v$ genutzt. Für Kantenoperationen werden analoge Notationen genutzt.

Definition 4.3
Gegeben seien zwei Graphen, der Quellgraph G und der Zielgraph H. Das Verfahren der Graph Edit-Distanz ist es, bestimmte Knoten und Kanten in G zu löschen, die übrigen Knoten und Kanten umzubenennen (zu substituieren) und eventuell Knoten und Kanten aus H einzufügen, so dass G in H transformiert wird. Eine Folge solcher Umformungsoperationen $o_1, ..., o_k$, die G komplett in H umwandelt, wird ein **Graph Edit-Path** (GEP) zwischen G und H genannt.

Für jedes Paar von Graphen (G, H) existieren unendlich viele verschiedene Graph Edit-Paths, die G in H transformieren. Das Löschen aller Kanten und Knoten aus G und das anschließende Einfügen aller Kanten und Knoten von H ist z.B. immer ein Graph Edit-Path, der G in H transformiert. Solche GEP sagen jedoch wenig über die Ähnlichkeit der beiden Graphen zueinander aus.
Generell können Substitutionen von Kanten und Knoten als positive Matches des Graphen G in den Graphen H bezeichnet werden, da die Benennungen ähnlich genug sind, um sie mit einer günstigen Umformungsoperation, der Substitution, umzuformen. Löschen und Einfügen von Kanten und Knoten bedeuten demzufolge,

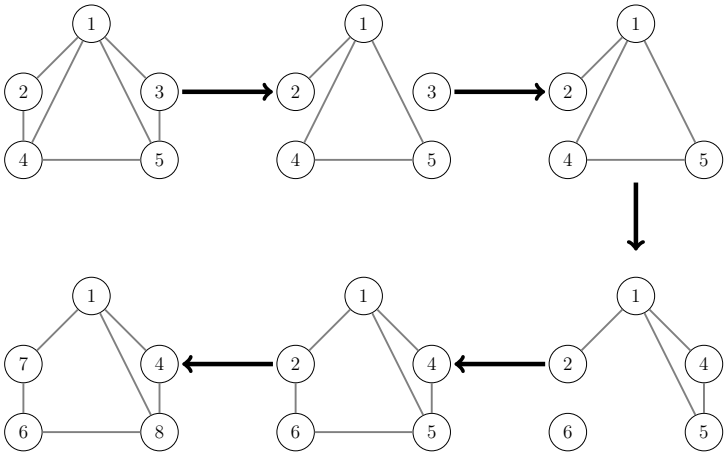

Abb. 4.2: Der Umformungsweg besteht aus drei Kantenlöschungen, einer Knotenlöschung, einer Knoteneinfügung, zwei Kanteneinfügungen und zwei Knotensubstitutionen.

dass für diese kein positives Matching gefunden wurde. Sind die Kosten für eine Substitution allerdings zu hoch, d.h. die ineinander zu transformierenden Kanten bzw. Knoten sind zu verschieden, werden diese Knoten und Kanten in G gelöscht bzw. aus H eingefügt. Mit Hilfe jedes Umformungswegs zwischen zwei Graphen G und H kann man Gemeinsamkeiten der Teilstrukturen der Graphen darstellen. In diesen Umformungswegen werden die Knoten/Kanten aus G entweder mit Knoten/Kanten aus H eindeutig substituiert oder gelöscht. Analog werden die Knoten/Kanten aus H entweder eindeutig mit Knoten aus G gematched oder eingefügt. Daher wird die Graph Edit-Distanz auch als Graph-Matching bezeichnet.
Sei $\Upsilon(G, H)$ die Menge aller Graph Edit-Paths zwischen G und H. Um einen aussagekräftigen Graph Edit-Path zu finden, werden die Kosten $c(o)$ für jede Umformungsoperation eingeführt. Diese sollen darstellen, wie stark der Einfluss einer Umformung auf die Struktur des Graphen ist. Die Kosten werden mit Hilfe von Graphinvarianten der einzelnen Knoten bzw. Kanten gebildet.

Beispiel:
Gegeben seien drei Knoten u, v, w mit den Knotengraden $\deg(u) = 1$, $\deg(v) = 2$ und $\deg(w) = 5$. Die Kosten, um u mit v zu substituieren sollen niedriger sein, als u mit w zu substituieren. Es gilt: $c(u \to v) < c(u \to w)$. Ebenso soll es günstiger sein, u mit v zu substituieren, als u zu löschen und v einzufügen. Es gilt: $c(u \to v) < c(u \to \epsilon) + c(\epsilon \to v)$.

Ziel der Kostenfunktion ist es, schwachen Veränderungen im Graph geringere Kosten zu geben und starken Umformungen hohe Kosten zu geben. Die **Kosten** eines Umformungswegs p zwischen G und H sei die Summe der Kosten für alle Umformungsoperationen von p und die **Länge** eines Umformungswegs p zwischen G und H sei die Anzahl seiner Umformungsoperationen. Zwischen zwei ähnlichen Graphen soll ein GEP mit niedrigen Kosten existieren und zwischen verschiedenen Graphen nur GEPs mit hohen Kosten.

Definition 4.4
Sei $G = (V, E)$ der Ursprungs- und $H = (W, F)$ der Zielgraph. Die **Graph Edit-Distanz** zwischen G und H sei

$$d(G, H) = \min_{(o_1,\ldots,o_k)\in\Upsilon(G,H)} \sum_{i=1}^{k} c(o_i) \tag{4.13}$$

mit $\Upsilon(G, H)$ der Menge aller Umformungswege, die G in H transformieren und c sei die Funktion, welche die Kosten $c(o)$ der Umformungsoperation o beschreibt.

Wenn weitere Informationen über die dem Graph zugrundeliegenden Objekte bekannt sind, können die Kostenfunktionen entsprechend angepasst werden, so dass z.B. das Löschen einer Kante teurer ist als das Löschen eines Knotens.
Jeder GEP in $\Upsilon(G, H)$ kann mit Hilfe zusätzlicher Einfügeoperationen und anschließender Löschoperationen beliebig verlängert werden. Damit mit Hilfe der Kostenfunktion nur wenige GEP aus $\Upsilon(G, H)$ betrachtet werden müssen, um die Graph Edit-Distanz zu bestimmen, muss die Kostenfunktion c folgende Eigenschaften erfüllen.

$$(1) \quad c(o) \geq 0$$

$$(2) \quad \begin{aligned} &c(u \to \epsilon) > 0 \\ &c(\epsilon \to v) > 0 \\ &c(e \to \epsilon) > 0 \\ &c(\epsilon \to e) > 0 \end{aligned}$$

Dadurch wird das Einfügen und anschließende Löschen von Kanten und Knoten, was

den GEP nur unnötig verlängert, verhindert. Als nächstes werden mit Hilfe folgender Dreiecksungleichungen unnötige Substitutionen vermieden.

$$
\begin{aligned}
(3) \quad & c(u \to w) \leq c(u \to v) + c(v \to w) \\
& c(u \to w) \leq c(u \to v) + c(v \to w) \\
& c(u \to \epsilon) \leq c(u \to v) + c(v \to \epsilon) \\
& c(\epsilon \to v) \leq c(\epsilon \to u) + c(u \to v)
\end{aligned}
$$

Mit den Eigenschaften (1), (2) und (3) können die zu betrachtenden Umformungswege in $\Upsilon(G, H)$ bereits stark reduziert werden. Jeder Umformungsweg, der eine unnötige Substitution oder das Einfügen und anschließende Löschen von Knoten und Kanten enthält, kann vernachlässigt werden, da ein kürzerer Umformungsweg mit niedrigeren Kosten existiert.
Gilt weiterhin, dass Substitutionen gleicher Objekte keine Kosten verursachen (4) und dass die Kostenfunktion symmetrisch ist (5), so ist die Graph Edit-Distanz eine Metrik.

$$
\begin{aligned}
(4) \quad & c(u \to u') = 0 \\
& c(e \to e') = 0
\end{aligned}
$$

$$
(5) \quad c(o) = c(o^{-}1)
$$

Normalerweise wird die Graph-Edit-Distanz mit Hilfe des *A*-Algorithmus* bestimmt, einem Verfahren zum Durchsuchen eines Graphen, bei dem in jeder Iteration der vielversprechendste Knoten gewählt wird. Dieser Knoten wird nach einer gewissen Heuristik bewertet. Der A*-Algorithmus ist damit ein Verfahren der Bestensuche (Best-first search) [BN07]. Der A*-Algorithmus ist optimal. d.h. er findet immer die optimale Lösung, falls eine existiert, hat allerdings eine exponentielle Laufzeit. Um die Edit-Distanz für große Graphen berechnen zu können, wird ein Näherungsalgorithmus mit polynomialer Laufzeit verwendet. Dieser berechnet jedoch nur eine suboptimale Lösung. Für weitere Informationen zur exakten Berechnung der GED siehe [BKL08].

4.3.4 Näherungsalgorithmus

Da der A*-Algorithmus zur Bestimmung der Graph Edit-Distanz eine exponentielle Laufzeit besitzt, wird versucht, dessen Laufzeit zu reduzieren bzw. die Verwendung

des A*-Algorithmus zu umgehen. Deshalb wurden u.a. Schätzungen für Schranken aufgestellt, die das Durchsuchen des Suchbaumes beschleunigen. Algorithmen für bestimmte Klassen von Graphen wurden ebenso erforscht, wie die Berechnung der GED mit Hilfe von Munkres Zuordnungsalgorithmus und einer quadratischen Formulierung der GED [BN07]. Da all diese Verfahren jedoch immer noch relativ hohe Laufzeiten besitzen bzw. nur auf bestimmte Klassen von Graphen angewandt werden können, wird in der vorliegenden Arbeit ein schneller Näherungsalgorithmus genutzt, der Umformungswege für Untergraphen nutzt. Die dafür genutzten Untergraphen sind die Nachbarschaftsuntergraphen der Knoten aus G und H. Diese sollen nun gematcht werden.

Die Grundidee des Näherungsalgorithmus ist, keine globale Bedingung zu erfüllen und so einen globalen optimalen Umformungsweg zu bestimmen. Stattdessen werden iterativ zwei Graphen miteinander verglichen, indem lokal Nachbarschaftsuntergraphen miteinander gematcht werden.

Ein Nachbarschaftsuntergraph besteht aus einem Knoten u, allen zu u adjazenten Knoten und den Kanten zwischen diesen (Definition 2.7.).

Eine Voraussetzung des Näherungsalgorithmus ist, das die Knoten der zu betrachtenden Graphen zweidimensionale Koordinaten besitzen, die ihre Position im $\mathbb{R}^2$ darstellen. Mit den Koordinaten können die Knoten eines Nachbarschaftsuntergaphen in eine geordnete Folge gebracht werden. Um die Folge zu erhalten, werden die Nachbarschaftsknoten mit Hilfe der Transformation der euklidischen Koordinaten in Polarkoordinaten im mathematisch positiven Sinn angeordnet. Anstatt

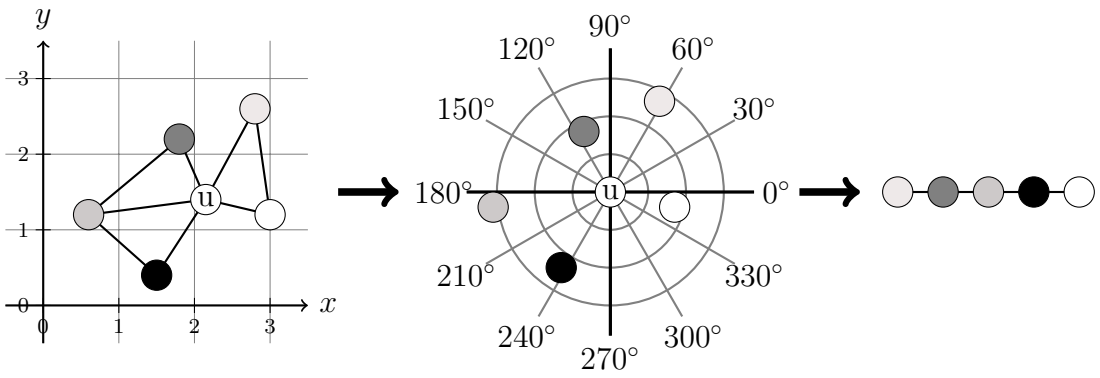

Abb. 4.3: Stringerstellung mit Hilfe der Koordinaten der Knoten aus $N(u)$ im $\mathbb{R}^2$

zwei Nachbarschaftsuntergraphen ineinander zu transformieren, wird eine optimale Angleichung der beiden Knotenfolgen gesucht. Im Gegensatz zum Matching zweier

Graphen, wird nun nur ein Umformungsweg gesucht, der die Ordnung der Knoten um den Zentralknoten bewahrt. Dazu wird das zyklische String-Matching genutzt [4.5].

Das zyklische String-Matching-Problem besitzt eine geringere Komplexität als der A*-Algorithmus und kann in quadratischer Laufzeit, in Abhängigkeit von der Stringlänge gelöst werden. Da bei der Angleichung zweier Knotenfolgen die Wahl des ersten Zeichens der Folge nicht von Bedeutung sein soll, wird ein zyklisches String-Matching anstelle eines normalen verwendet.

Die String-Umformungskosten für Löschen, Einfügen oder Substituieren eines Zei-

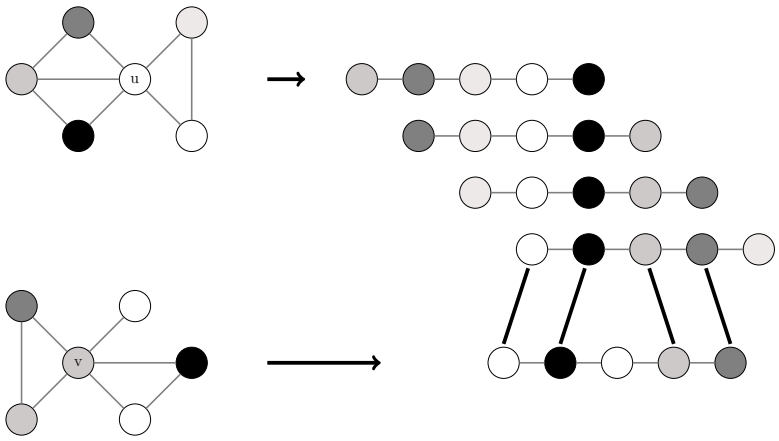

Abb. 4.4: Beispiel für das Matching zweier Nachbarschaften $N(u)$ und $N(v)$, mit 4 substituierten Knoten, einem gelöschten und einem eingefügten Knoten

chens, d.h. Knotens, lassen sich von den zugehörigen Umformungskosten der Knoten- und Kantenoperationen ableiten. Der aus dem String-Matchingalgorithmus resultierende Umformungsweg mit Minimalkosten liefert den optimalen Weg, um einen Nachbarschaftsuntergraphen in den anderen zu transformieren. Der Näherungsalgorithmus ist ein sogenannter GREEDY-Algorithmus, d.h. es werden nur Umformungskosten addiert, aber nie wieder abgezogen. Daher ist der aus dem Näherungsalgorithmus resultierende Graph Edit-Path p suboptimal. Weiterhin ist die mit dem Näherungsalgorithmus berechnete Graph Edit-Distanz kein metrisches Ähnlichkeitsmaß, da sie die Bedingung der Symmetrie nicht erfüllt.

Algorithmus 4.3.1: Algorithmus zur Bestimmung der suboptimalen GED

Eingabe: Zwei nichtleere zusammenhängende Graphen $G = (V, E)$, $H = (W, F)$ mit $V = \{v_1, ..., v_{|V|}\}$ und $W = \{w_1, ..., w_{|W|}\}$.

Ausgabe: Die suboptimale GED, samt Umformungsweg p.

Initialisierung der leeren Menge p;

Initialisierung der leeren first-in-first-out Schlange OPEN;

Bestimmung der Graphzentren v_{i_0} und w_{i_0} mit Netzwerkzentralitäten;

Hinzufügung der Startsubstitution $v_{i_0} \rightarrow w_{i_0}$ in p und OPEN;

while *OPEN ist nichtleer* **do**

  Wahl der nächsten Substitution $v \rightarrow w$ aus OPEN;

  Erzeugung des Nachbarschaftsuntergraphen $N(v)$ und $N(w)$ aus G und H;

  Erstellung der geordnete Zeichenfolgen von Knoten aus $N(v)$ und $N(w)$;

  Matching der beiden Zeichenfolgen mit zyklischem String-Matching;

  q sei der daraus resultierende Umformungsweg zwischen $N(v)$ und $N(w)$;

  Einfügen aller Umformungsoperationen aus q in p;

  Einfügen alle Knotensubstitutionen aus q in OPEN;

end

for *alle ungematchten Knoten* v_j *aus* G **do**

  Hinzufügen von $u_j \rightarrow \epsilon$ in p;

end

for *alle ungematchten Knoten* w_j *aus* H **do**

  Hinzufügen von $\epsilon \rightarrow w_j$ in p;

end

4.3.5 Einbettung der Graphen

Um den Näherungsalgorithmus für die Berechnung der GED anwenden zu können, müssen die dazu genutzten Graphen in den $\mathbb{R}^2$ eingebettet sein. Für die Einbettung wurde der Algorithmus von Kamada und Kawai genutzt. Die grundlegende Idee dieses Algorithmus ist es, dass die Distanz zweier Knoten innerhalb eines Graphen die gewünschte euklidische Distanz im $\mathbb{R}^2$ darstellt. Dazu wird u.a. der Algorithmus von Dijkstra zur Bestimmung der kürzesten Wege innerhalb eines Graphen und ein spring-embedding Verfahren genutzt. Dieser nutzt physikalische Gesetzmäßigkeiten und das Verhalten von Federn, um den Graphen einzubetten. Ziel dieser Einbettung ist es nicht, den Graphen möglichst planar einzubetten, d.h. wenig sich kreuzende Kanten zu besitzen. Durch die Einbettung sollen ähnliche Strukturen ähnlich dargestellt werden, so dass der Nutzer selbst diese ähnlichen Strukturen in der Einbettung erkennt. Der Graph embedder besitzt eine kubische Laufzeit und ist nicht deterministisch.

Weiterführende Informationen zur Einbettung von Graphen in den $\mathbb{R}^2$ finden sich in [KK89].

4.4 Levenshteindistanz

Der Levenshtein-Algorithmus errechnet die Mindestanzahl von Editierungsoperationen, die notwendig sind, um eine bestimmte Zeichenkette u soweit abzuändern, um eine andere bestimmte Zeichenkette v zu erhalten. Diese Mindestanzahl wird als *Levenshteindistanz* bezeichnet und ist nach dem russischen Wissenschaftler Wladimir Löwenstein benannt, der sie 1965 einführte. Die Levenshtein-Distanz ist eine Metrik auf dem Raum der Zeichenketten, d.h. sie erfüllt die Dreiecksungleichung, ist symmetrisch und reflexiv [Lev65].
Die Berechnung der Levenshtein-Distanz (auch Edit-Distanz genannt) erfolgt durch den sogenannten Dynamic-Programming-Ansatz. Dabei wird eine Distanzmatrix $D_{m\times n}$ für die beiden Zeichenketten u und v rekursiv aufgestellt:

$$
\begin{aligned}
m = |u|,\, n = |v| \\
D_{0,0} = 0 \\
D_{i,0} = i, 1 \leq i \leq m \\
D_{0,j} = j, 1 \leq j \leq n \\
D_{i,j} = \min \begin{cases} D_{i-1,j-1}, & +0 \text{ falls } u_i = v_j,\ (\mathbf{E}), \\ D_{i-1,j-1}, & +1 \text{ (Substitution)}, (\mathbf{S}), \\ D_{i,j-1}, & +1 \text{ (Einfügung)}, (\mathbf{I}), \\ D_{i-1,j}, & +1 \text{ (Löschung)}, (\mathbf{D}), \end{cases}
\end{aligned}
$$

wobei $1 \leq i \leq m, 1 \leq j \leq n$. Die Levenshtein-Distanz, d.h. die minimale Anzahl der Edit-Operationen ist gleich $D_{m,n}$.
Wenn nicht nur die Levenshtein-Distanz berechnet werden soll, sondern auch die Folge der Operationen, die zu dem Ergebnis geführt hat, muss ein Backtrace auf der berechneten Matrix durchgeführt werden. Beginnend von $D_{i,j-1}$ werden rekursiv die Optimierungsentscheidungen zurückverfolgt. Das Backtrackverfahren wird in abgeänderter Form für das zyklische String-Matching verwendet.
Folgend soll als Beispiel die Zeichenketten *'Bachelor'* in *'Master'* transformiert werden (Bachelor→Master). Damit ergibt sich in Tab. 4.1 eine Edit-Distanz von

		M	a	s	t	e	r
	0	1	2	3	4	5	6
B	1	1	2	3	4	5	6
a	2	2	1	2	3	4	5
c	3	3	2	2	3	4	5
h	4	4	3	3	3	4	5
e	5	5	4	4	4	3	4
l	6	6	5	5	5	4	4
o	7	7	6	6	6	5	5
r	8	8	7	7	7	6	5

Tab. 4.1: Levenshteindistanz für Bachelor→Master

5, der Umformungsweg samt Operationen lassen sich aus Tab. 4.2 entnehmen. Die

S	E	S	S	E	D	D	E
B	a	c	h	e	l	o	r
M	a	s	t	e			r

Tab. 4.2: Der Umformungsweg für Bachelor→Master

Levenshteindistanz lässt sich auch auf andere Distanzen überführen. Sind z.B. nur Substitutionen bei der Transformation gleichlanger Zeichenketten erlaubt, erhält man die Hammingdistanz. Diese gibt genau an, an wie vielen Positionen sich zwei Zeichenketten unterscheiden.
In der Praxis wird die Levenshtein-Distanz zur Bestimmung der Ähnlichkeit von Zeichenketten, beispielsweise zur Duplikaterkennung oder bei der Rechtschreibprüfung von Suchmaschinen angewandt.

4.5 Das zyklische String-Matching

Die Levenshteindistanz lässt sich mit wenigen Änderungen auch auf zyklisch verschobene Zeichenketten anwenden. Dieses zyklische String-Matching wird zur suboptimalen Berechnung der GED verwendet. Die folgenden Ausführungen sind an [BB93] angelehnt.
Das zyklische String-Matching ist eine Brute Force-Methode, um zu bestimmen, ob die Zeichenkette $x^{(i)} = x_i...x_n x_1...x_{i-1}$ eine rotierte Version einer der Prototypen $y = y_1...y_n$ bzw. $y^{(i)} = y_i...y_n y_1...y_{i-1}$ für $i \in \{1, .., n\}$ ist [BB93]. Dafür wird die Zeichenkette $yr = yy = y_1...y_n y_1...y_n$ erzeugt und es wird ein Umformungsweg von x

in eine Teilzeichenkette von y^2 gesucht, unabhängig vom Startpunkt i dieser Teilzeichenkette in y^2.
Um festzustellen, ob x eine Teilzeichenkette von y^2 ist, wird eine modifizierte Version des Algorithmus von Wagner und Fischer [WF74] genutzt, um eine Distanzmatrix für das String-Matching von x in y^2 aufzustellen. Die dabei verwendete Distanz ist die *Levenshteindistanz*. In der Distanzmatrix $D(i, j)$ wird nun nach dem kleinsten Element in $D(n, j) = n, ..., 2n - 1$ gesucht. Ist dieses Element gleich null, so ist x eine Teilzeichenkette von y^2 bzw. ist y eine zyklisch verschobene Version von x. Normalerweise ist y keine zyklisch verschobene Version von x, y ist zusätzlich durch Substitutionen, Einfügungen und Löschungen verformt.
Der Algorithmus des zyklischen String-Matching bestimmt für zwei Zeichenketten $x = x_1...x_n$ und $y = y_1....y_m$ die Edit-Distanz $d(x, y')$, wobei $y' = y_i...y_m y_1...y_j$ mit $1 \leq j \leq i \leq m$ die Teilzeichenkette aus y^2 ist, die x am ähnlichsten ist.
Die Edit-Distanz ergibt sich aus:

$$d(x, y') = \min\{d(x, z) | \text{z ist Teilzeichenkette von } y^2\}. \tag{4.14}$$

Die Kosten für die Umformungsoperationen für Zeichen werden bei der Berechnung der suboptimalen GED durch die Kosten der entsprechenden Umformungsoperationen für Kanten und Knoten aufgestellt. Der Algorithmus hat eine Laufzeit von $\mathcal{O}(n \cdot m)$, wobei n die Länge der Zeichenkette x und m die Länge der Zeichenkette y ist.

Algorithmus 4.5.1: Algorithmus zyklisches String-Matching

Eingabe: Zwei Zeichenketten $x = x_1...x_n$ und $y = y_1...y_m$ und die Kostenfunktion c für jede Umformungsoperation o.

Ausgabe: Die zyklische Edit-Distanz, samt Umformungsweg p.

Erstellung von y^2 und Initialisierung der leeren Menge p;

Initialisiere die Distanzmatrix D;

$D(0,0) := 0$;

for $i = 1, ..., m$ **do**

  $D(0,i) := 0$;

end

for $i = m+1, ..., 2m$ **do**

  $D(0,i) := \infty$;

end

for $i = 1, ..., n$ **do**

  $D(i,0) := D(i-1,0) + c(x_i \to \epsilon)$;

end

Berechnung die restliche Einträge von D;

for $i = 1, ..., n$ **do**

  for $j = 1, ..., 2m$ **do**

    $m_1 := D(i-1,j-1) + c(x_i \to y_j)$,;

    $m_2 := D(i-1,j) + c(x_i \to \epsilon)$,;

    $m_3 := D(i,j-1) + c(\epsilon \to y_j)$,;

    $D(i,j) = \min(m_1, m_2, m_3)$,;

  end

end

Bestimmung von $d(x,y) = D(n,k)$ mit $D(n,k) = \min\{D(n,j) | j = n, ..., 2m\}$;

Ermittlung des Umformungswegs in D über ein Backtracking der Matrix;

for $i = n, ..., 1$ **do**

  for $j = k, ..., 1$ **do**

    if $D(i-1,j-1) + c(x_i \to y_j) = D(i,j)$ **then**

      Hinzufügen von $c(x_i \to y_j)$ in p;

    else

      if $D(i-1,j) + c(x_i \to \epsilon) = D(i,j)$ **then**

        Hinzufügen von $c(x_i \to \epsilon)$ in p;

      else

        if $D(i,j-1) + c(\epsilon \to y_j) = D(i,j)$ **then**

          Hinzufügen von $c(\epsilon \to y_j)$ in p;

        end

      end

    end

    $D(i,0) := D(i-1,0) + c(x_i \to \epsilon)$;

  end

end

Ausgabe von p mit den Kosten $d(x,y)$;

5 Evaluierung der Ergebnisse

Das folgende Kapitel soll einen Einblick auf die Nutzung des Verfahrens geben. Dabei teilt sich die Anwendung in zwei Teile auf. Zuerst soll der Graph in entsprechend vergleichbare Prototypen partitioniert werden. Dies geschieht mit Hilfe der spektralen Bisektion bzw. Multisektion aus 4.1 bzw. 4.2. Der zweite Teil umfasst das Vergleichen der Graphen mit der *Graph Edit-Distanz* aus 4.3 und der darauf aufbauenden Bestimmung der repräsentativen Prototypen [3.2]. Die hier untersuchten Daten stammen von einem Chip, auf dessen Platine 16 Prozessoren miteinander verdrahtet wurden. Daraus ergab sich ein Graph G mit $n = 97937$ Knoten, der nun auf ähnliche Grundstrukturen, d.h. repräsentative Prototypen, untersucht werden soll.

5.1 Partitionierung des Graphen

Um entsprechend kleine Prototypen zu erhalten, wird der Graph nun mit dem Algorithmus, welcher in 4.2 beschrieben wird, hierarchisch partitioniert. Heuristisch wurde hierbei die Größe der zu betrachtenden Teilgraphen auf 100 bis 150 Knoten geschätzt. Entsprechend große Blöcke V_i wurden mit in den nächsten Partitionierungsschritt übernommen, ohne weiter zerlegt zu werden. Hierfür wurde zunächst die spektrale Multisektion verwendet. Die Anzahl der dabei zu untersuchenden Cluster wurde mit Hilfe der Eigengap-Heuristik bestimmt. Allerdings entstanden bei diesem Verfahren innerhalb der Partitionierungen zu viele isolierte Knoten, d.h. Knoten vom Grade null. Tab. 5.1 soll bis zum 3. Partitionierungsschritt (PS) einen Einblick auf die Anzahl und Beschaffenheit der Blöcke und der entstandenen isolierten Knoten geben. Hierbei sei $|\dot{n}(P)|$ die Anzahl der isolierten Knoten.
Die isolierten Knoten mussten nach jedem Partitionierungsschritt aus den Teilgraphen entfernt werden, da sie die Eigengap-Heuristik beeinflussen. Für jeden isolierten Kno-

PS	1	2	3
$\lvert P \rvert$	17	125	220
$\sum_{\substack{V_i \in P, \\ \lvert V_i \rvert < 100}} V_i$	0	0	11
$\sum_{\substack{V_i \in P, \\ \lvert V_i \rvert \in [100,150]}} V_i$	0	16	29
$\sum_{\substack{V_i \in P, \\ \lvert V_i \rvert > 150 \\ Eigengap(V_i)=1}} V_i$	1	26	66
$\lvert \dot{n}(P) \rvert$	75	295	562

Tab. 5.1: Verlauf der spektralen Multisektion.

ten, d.h. zusätzliche Komponente eines Teilgraphen, steigt dessen Eigengap um eins. Dies führt anschließend zu schlechten Partitionierungen, da der k-Means-Algorithmus sphärisch partitioniert und dem zusätzliche Cluster, der eigentlich nur aus einem Knoten besteht, werden mehrere Knoten zugeordnet. Dadurch werden auch die restlichen Cluster und damit die restliche Partitionierung nachteilig beeinflusst.
Die Anwendung der Eigengap-Heuristik sorgt für weitere Probleme. So ergaben sich schon in der zweiten Partitionsstufe, unabhängig von der Größe der Teilgraphen, viele Eigengaps gleich eins. Diese Teilgraphen, die mehrfach nicht aus den angestrebten 100 bis 200 Knoten bestanden, lassen sich gemäß Spektraltheorie nicht weiter zerlegen. Erst die Verwendung des zweitgrößten Eigengaps machte die weitere Zerlegung dieser Teilgraphen möglich. Dieser betrug jedoch meist nur ein Achtel des eigentlichen Eigengaps und resultiert in Partitionen mit isolierten Knoten. Umgekehrt gab es jedoch relativ kleine Teilgraphen, die gemäß ihres Eigengaps in zu viele, zu kleine Teilgraphen zerlegt worden wären, deren Größe sich aber auch noch nicht innerhalb des angestrebten Intervalls befand. Deshalb wurde die hierarchische Partitionierung bereits im 4. Schritt abgebrochen und der Algorithmus aus 4.1 eingesetzt. Anzumerken ist hierbei, dass die Eigengap-Heuristik im ersten Schritt für G bei 17 lag, was den 16 Prozessoren und einer Hauptleitung entsprechen würde.
Die spektrale Bisektion sollte mit Hilfe des Median und einer vorgegebenen Partitionierungsstufe den Graphen G in eine bestimmte Anzahl an Teilgraphen vergleichbarer Größe zerlegen. Angestrebt ist diesmal eine Teilgraphengröße von 100 bis 200. Der Median teilt die Blöcke einer Partitionierung nicht immer gleich groß, z.B. wenn der Median als Element innerhalb eines Vektors mehrfach vorkommt. So ergaben sich für die 512 Teilgraphen des 10. Partitionierungsschrittes ein Minimum von $n = 158$

PS	1	2	3	4	5	6	7	8	9
$\|P\|$	2	4	8	16	32	64	128	256	512
$\|\dot{n}(P)\|$	1579	4388	4415	5217	5878	14787	21218	25670	28517

Tab. 5.2: Entwicklung der Anzahl isolierter Knoten während der spektralen Bisektion

bzw. ein Maximum von $n = 332$. Wie in Tab. 5.2 zu erkennen, ergaben sich viele isolierte Knoten während der einzelnen Partitionierungsschritte, bis im letzten Schritt 28517 Knoten isoliert waren, was ungefähr 29 Prozent der Knoten von G entspricht. Da diese Knoten nicht zur Erzeugung von Prototypen genutzt werden können, geht damit ein beachtlicher Teil der strukturellen Information von G verloren. Zusätzlich zu der großen Anzahl isolierter Knoten sind viele der entstandenen Teilgraphen nicht zusammenhängend.
Anschließend wurde die spektrale Multisektion mit 2 Clustern verwendet. Dadurch ging die Kontrolle über die Anzahl bzw. Größe der entstehenden Teilgraphen verloren. Die Bedingung, dass kein Teilgraph mehr als 150 Knoten enthalten soll und die Wahl von $k = 2$, soll die hierarchische Partitionierung dabei ein wenig steuern, so dass Teilgraphen vergleichbarer Größe entstehen sollen.

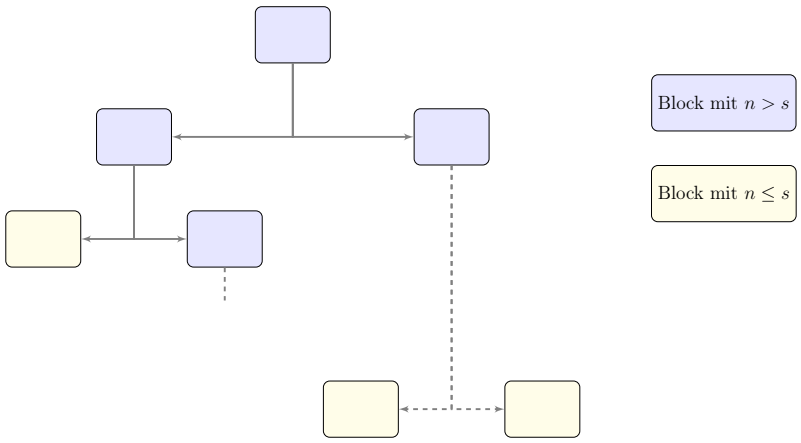

Abb. 5.1: Spektrale Multisektion in zwei Teilgraphen ohne Größenvorgabe

Mit Hilfe der spektralen Multisektion mit $k = 2$ wurde der Graph G solange hierarchisch partitioniert, bis alle 982 daraus entstandenen Partitionen eine maximale Größe von 150 Knoten hatten. Weiterhin sind die entstandenen Teilgraphen zusammenhängend, d.h. sie besitzen auch keine isolierten Knoten. Während der Partitionierung konnte kein Einfluss auf die Größe der entstehenden Blöcke genommen werden,

Testklasse	150	300	500	700
$0 < n \leq 50$	32	-	-	-
$50 < n \leq 100$	480	22	4	2
$100 < n \leq 150$	470	97	23	8
$150 < n \leq 200$	-	150	26	12
$200 < n \leq 250$	-	146	47	7
$250 < n \leq 300$	-	89	55	13
$300 < n \leq 350$	-	-	60	24
$350 < n \leq 400$	-	-	43	17
$400 < n \leq 450$	-	-	42	29
$450 < n \leq 500$	-	-	22	16
$500 < n \leq 550$	-	-	-	23
$550 < n \leq 600$	-	-	-	27
$600 < n \leq 650$	-	-	-	17
$650 < n \leq 700$	-	-	-	25
$\sum$	982	504	322	220

Tab. 5.3: Größenverteilung innerhalb der Testklassen

sodass z.T. relativ kleine Blöcke (z.B. mit $n = 11$) entstanden. Die genauen Größen können Tab. 5.3 für die Klasse 150 entnommen werden. Da die GED allerdings in Abhängigkeit einer Kostenfunktion, die Rücksicht auf die Größe beider Teilgraphen nimmt, bestimmt wird, können die GED zweier Graphen mit gleicher Knotenanzahl mit der GED zweier Graphen, für die gilt $n_1 << n_2$, verglichen werden.
Durch die hierarchische Partitionierung entstanden für die einzelnen Partitionierungsschritte Teilgraphen verschiedener Größen, die u.a. weiter zerlegt wurden. Die Teilgraphen verschiedener Schritte werden nun in Testklassen zusammengefasst, deren Vereinigung gleich der Knotenmenge von G entspricht. Weiterhin sollen die Teilgraphen einer Testklasse disjunkt und möglichst groß sein, ohne dabei eine bestimmte Anzahl an Knoten zu übersteigen.
Somit ergeben sich die vier Testklassen, deren Teilgraphen aus maximal $150, 300, 500$ bzw. 700 Knoten bestehen. Die Idee hinter dieser Einteilung ist es, relativ gleich große Graphen miteinander zu vergleichen.
Die Testklassen wurden nun auf die tatsächlichen Größenverhältnisse untersucht. Dafür wurden die Teilgraphen ihrer Größe nach in Tab. 5.3 in Gruppen eingeteilt.

5.2 Parameter der suboptimalen GED

Bei der Berechnung der suboptimalen GED können mehrere Parameter eingestellt werden, die starken Einfluss auf das resultierende Ähnlichkeitsmaß haben. Die Ähnlichkeit zweier Graphen hängt teilweise stark von der richtigen Einstellung, z.B. der zugrunde liegenden Kostenfunktion ab. Da für die Wahl der Parameter keine Erfahrungen vorliegen, müssen diese mehr oder weniger empirisch bestimmt werden.
Zum Vergleich der einzelnen Parametereinstellungen werden die Verfahrensvarianten wie folgt benannt: *GED_PC_PO_PF_PS*. Hierbei bezeichnet

PC ... die Bestimmung des Zentrums,

PO ... die Anordnung der Nachbarschaften,

PF ... die verwendete Kostenfunktion,

PS ... die Art der Ähnlichkeitsmatrix.

So ergibt sich für ein Verfahren, das zur Bestimmung des Zentrums die Eigenwertzentralität nutzt, für die Ordnung der Nachbarschaft als primäres Merkmal ebenfalls die Eigenwertzentralität und als sekundäres Merkmal die Knotenexzentrizitäten verwendet, die Kosten mit der Standardkostenfunktion berechnet und die Ähnlichkeitsmatrix mit den Graph Edit-Distanzen aufstellt die Bezeichnung *GED_E_EM_CC1_D*. Ein Verfahren, welches die Graph Edit-Distanz ebenso berechnet, die Ähnlichkeitsmatrix aber mit der GEP aufstellt, wird mit *GED_E_EM_CC1_P* bezeichnet.

5.2.1 Zentrumswahl PC und Ordnung der Nachbarschaft PO

Die Wahl des Zentrums gibt an, in welchem Knoten die Suboptimale GED beginnt die Graphen miteinander zu vergleichen. Zur Bestimmung des Startpunktes wird der Knoten gewählt,

- mit den Knotenexzentrizitäten (und dem zugehörigen Zentrum) (**M**),
- mit größtem Knotengrad (**D**),
- mit der größten Eigenwertzentralität (**E**),

Nach der Wahl des Zentrums ist die Anordnung der Nachbarschaft ein weiterer Parameter, der das Verhalten des Algorithmus entscheidend steuert. Ursprünglich wurden dafür nur die Koordinaten einer Einbettung der Graphen genutzt **(C)**. Diese

	Var 1	Var 2	Var 3	Var 4
Var 1	0.0000	0.5052	0.0000	0.0000
Var 2	0.5052	0.0000	0.5052	0.5052
Var 3	0.0000	0.5052	0.0000	0.0000
Var 4	0.0000	0.5052	0.0000	0.0000

Tab. 5.4: Vergleich der Einbettungen mit *GED_E_C_CC1_D*

Einbettungen sind jedoch nicht deterministisch [4.3.5]. Diese nachteilige Eigenschaft überträgt sich hierbei auf das resultierende Ähnlichkeitsmaß und kann dieses stark verzerren.
Um dies zu verdeutlichen, werden für vier zufällige Einbettungen eines Graphen die suboptimalen Graph Edit-Distanzen bestimmt.

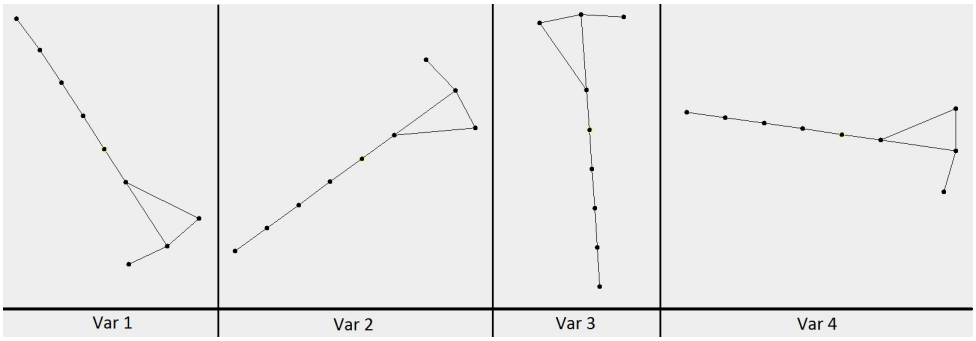

Abb. 5.2: Vier verschiedene Einbettungen von G

Hier ist in Tab. 5.4 deutlich zu erkennen, dass sich für den Graphen mit der Einbettungsvariante Var 2 von null verschiedene suboptimale GED ergeben, obwohl alle vier Graphen genau die gleichen strukturellen Eigenschaften besitzen. Vergleicht man die 4 eingebetteten Graphen mit einer suboptimalen GED, deren Nachbarschaften nicht über die Einbettungen geordnet werden, ergibt sich das korrekte Ähnlichkeitsmaß von null zwischen allen Graphen.
Um solche Verzerrungen zu vermeiden, wurden nun andere Anordnungsmöglichkeiten für die Nachbarschaft gesucht. Dafür werden mit Hilfe der Netzwerkzentralitäten, die schon zur Bestimmung des Zentrums verwendet wurden, die Knoten auf- bzw. absteigend angeordnet. Bei gleichen Zentralitäten wird eine von der ersten verschiedene Netzwerkzentralität zur Ordnung genutzt. In der Auswertung werden folgend nur eine Auswahl an Kombinationen vorgestellt, da die Auswahl des zweiten Ordnungsmerkmals meist kaum Auswirkungen auf die GEPs haben.

5.2.2 Wahl der Kostenfunktion PF

Ebenso wichtig für aussagekräftige Ähnlichkeitsmaße ist die Wahl einer geeigneten Kostenfunktion [Bun99], welche die Wichtigkeit bestimmter Operationen darstellen soll [4.3.3]. Die Kosten für die einzelnen Graphoperationen werden mit Hilfe der gewichteten Hilfskonstanten $\alpha_{G_1}, \alpha_{G_2}, \beta_{G_1}, \beta_{G_2}$ berechnet. Diese können mit Hilfe der Kanten- und Knotenmenge der Graphen z.B. wie folgt berechnet werden:

$$\alpha_G = w_n \times \frac{1}{n_G} = \frac{1}{4 \times n_G},$$
$$\beta_G = w_e \times \frac{1}{m_G} = \frac{1}{8 \times m_G}.$$

Berechnungen der Hilfskonstanten über den Vergleich der Größen der Graphen sind ebenfalls möglich, jedoch bei großen Größenunterschieden weniger aussagekräftig. Die Gewichte w_n und w_e geben dabei die Wichtigkeit einer Knoten- bzw. Kantenoperation wieder. Für die Gewichte soll gelten:

$$\alpha_{G_1} + \alpha_{G_2} + 2\beta_{G_1} + 2\beta_{G_2} = 1.$$

Werden alle Knoten und Kanten aus G_1 gelöscht und alle Knoten und Kanten aus G_2 eingefügt, ergibt sich eine $GED = 1$, werden keine Operationen ausgeführt, ist die $GED = 0$. Werden Kanten- und Knotenoperationen gleich gewertet, ergeben sich folgende Gewichte: $w_n = \frac{1}{4}, w_e = \frac{1}{8}$. Damit ergeben sich für die Kostenfunktion **CC1** für die Knotenoperationen folgende Kosten:

$$c_{nd} = \alpha_{G_1} + \beta_{G_1} \times \deg(v),$$

$$c_{ni} = \alpha_{G_2} + \beta_{G_2} \times \deg(u),$$

$$c_{st} = \begin{cases} \beta_{G_1} \times |\deg(v) - \deg(u)|, & \deg(v) - \deg(u) \geq 0 \\ \beta_{G_2} \times |\deg(v) - \deg(u)|, & \text{sonst.} \end{cases}$$

Bemerkung:
Da die Knotengrade innerhalb der Nachbarschaftsuntergraphen kleiner sind als in den Ursprungsgraphen, folgt dass in der suboptimalen GED das Entfernen und Einfügen von Knoten innerhalb der Nachbarschaftsuntergraphen billiger ist als die Anwendung

der gleichen Operationen auf die Knoten in G_1 bzw. G_2 am Ende des Algorithmus. Da die Berechnung der suboptimale GED kein Gedächtnis besitzt, können Knoten innerhalb mehrerer Nachbarschaftsumwandlungen z.B. gelöscht werden, was sich negativ auf die GED auswirkt. Kleine Graphen, die nur wenige Nachbarschaften matchen müssen, besitzen demnach weniger Kosten für wiederholte Operationen an einem Knoten, größere Graphen entsprechend mehr. Dadurch wird die GED weniger aussagekräftig. Dem kann jedoch entgegen gewirkt werden, indem die GEPs betrachtet werden, welche keine mehrfachen Operationen enthalten.

5.2.3 Aufstellen der Ähnlichkeitsmatrix PS

Eine Unähnlichkeitsmatrix für Graphen enthält die GED zwischen den Graphen als Einträge **D**. Weiterhin gilt für die Elemente der Unähnlichkeitsmatrix $\overline{S}$: $\overline{s_{i,j}} \in [0,1]$. Die Transformation in eine Ähnlichkeitsmatrix S erfolgt mit Hilfe eines Grenzwertes ε, der z.B. über den Durchschnitt oder den Median der Unähnlichkeitsmatrix berechnet werden kann.

$$s_{i,j} = \begin{cases} 1 - \overline{s_{i,j}}, & 1 - \varepsilon >= \overline{s_{i,j}}, \\ 0, & sonst. \end{cases} \tag{5.1}$$

Dadurch wird S zur Adjazenzmatrix eines ε-Nachbarschaftsgraphen, mit den Prototypen als Knoten und den größten Ähnlichkeiten als gewichteten Kanten [Lux07]. Diese Kantenreduktion kann iterativ wiederholt werden, um den Graphen besser partitionieren zu können. Die besten Ergebnisse wurden hierbei bei einer Iterationstiefe von 3 bzw. 4 mit Hilfe des Median erreicht. Für niedrigere Iterationsdichten ergaben sich im Durchschnitt zu wenige, sehr große Cluster. Für diese kann kaum ein repräsentativer Prototyp bestimmt werden, da dieser zu vielen Graph Edit-Paths entsprechen müsste. Für höhere Iterationsdichten ergaben sich allerdings zu viele Cluster innerhalb einer Partitionierung, die zu großen Teilen nur aus einem oder zwei Teilgraphen bestanden, die nicht zur Generierung eines Prototypen eingesetzt werden können. Folgende Tabelle soll dies verdeutlichen, indem die Anzahl der Cluster aus der Testklasse 150 mit 982 Teilgraphen, die durch verschiedene Verfahrensvarianten erzeugt wurden, miteinander verglichen werden. Dabei ist zu erkennen, dass die höheren Reduktionsstufen zu einer Clusteranzahl führt, die der Größe der Testklasse entspricht, d.h. jeder Cluster besteht nur aus einem Teilgraphen. Deshalb werden

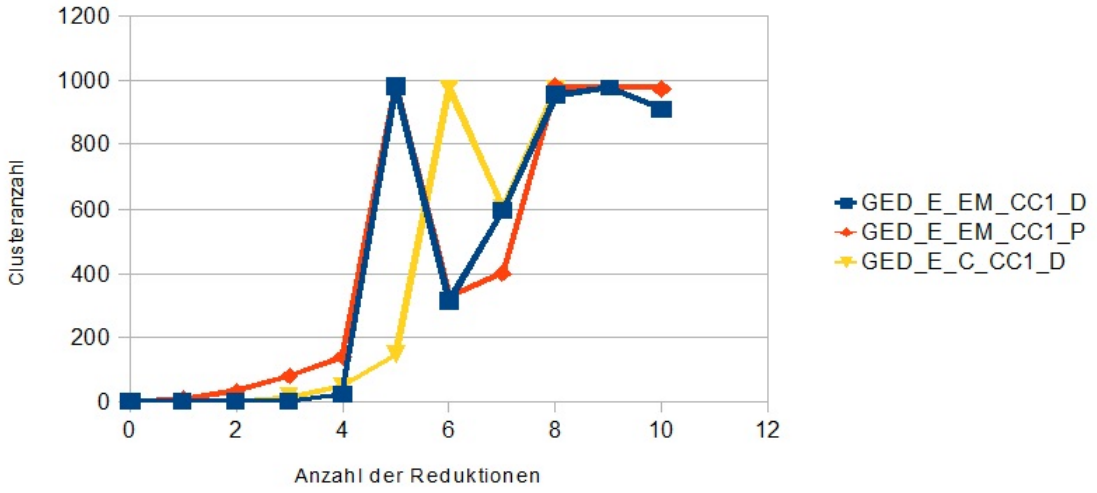

Abb. 5.3: Verhältnis der Reduktionsstufen zur Clusteranzahl

folgende nur die ersten 5 Reduktionsstufen der Ähnlichkeitsmatrizen betrachtet, da diese die vielversprechendsten Partitionierungen enthalten.
Statt der GED kann allerdings auch die GEP zur Aufstellung der Ähnlichkeitsmatrix genutzt werden **P**. Die Matrix enthält statt der GED zwischen zwei Teilgraphen die Anzahl der Operationen, um einen Graphen in den anderen umzuwandeln. Die besten Ergebnisse wurden hierbei erzielt, wenn nur die Anzahl der Substitutionen übernommen werden. Anschließend werden die Matrixelemente wieder mit Hilfe des Median reduziert. Die dadurch erreichten Partitionierungen sind weitaus kompakter als die vergleichbaren Partitionierungen mit Hilfe der GED.

5.3 Bestimmung der Prototypen und graphische Darstellung

Zur Bestimmung des Prototypen P des Clusters C mit n Teilgraphen wird zunächst der dem Clusterzentrum nächstliegende Teilgraph T ausgewählt. Dazu wird zunächst eine Ähnlichkeitsmatrix für C aufgestellt, die nur Einträge für die n Cluster aus C enthält. Für die aufsummierte Summe der GEDs von T innerhalb des Clusters muss gelten:

$$\sum_{i=1}^{n} s_{i,k} = \max\{\sum_{i=1}^{n} s_{i,j}, j = 1, .., n\},$$

die k-te Zeile enthält hierbei den Teilgraphen T.
Auf Grundlage der Knoten und Kantenmengen von T wird nun der Prototyp des Clusters generiert, d.h. es gilt $V(P) \subseteq V(T)$ und $E(P) \subseteq E(T)$. Der Prototyp P ist somit ein Teilgraph von T. Zur Bestimmung der Kanten- und Knotenmenge von P werden die Umformungswege von T zu den restlichen Teilgraphen aus C bestimmt. Anschließend werden die Operationen bzw. die aus ihnen entstandenen Kosten aufsummiert. Sollten Knoten aus T dabei eine vorgegebene Kostengrenze überschreiten, werden sie und alle inzidenten Kanten aus $V(P)$ bzw. $E(P)$ gelöscht. Zur Veranschaulichung werden die Prototypen des Clusters eines Graphen anschließend gezeichnet. Hierbei werden die Knoten entsprechend ihrer aufsummierten Kosten eingefärbt, um weiter zu verdeutlichen, welcher Knoten mehr Kosten bei der Umformung des Prototypen in die restlichen Graphen aus C erzeugt. Knoten mit geringen Kosten werden hierbei dunkler gezeichnet, als Knoten mit höheren Kosten. Überschreitet ein Knoten die Kostengrenze, wird er samt allen inzidenten Kanten vom restlichen Graphen entfernt.

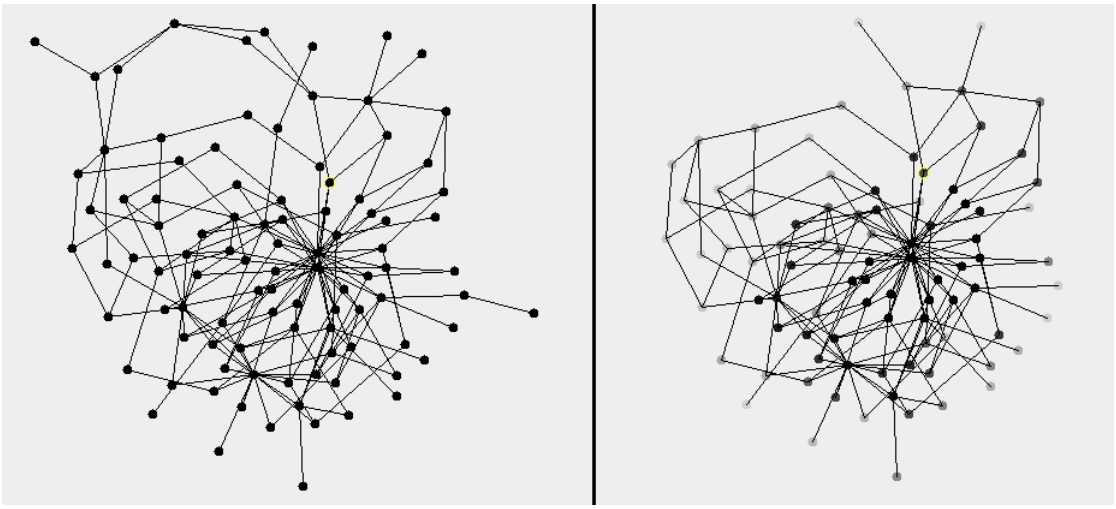

Abb. 5.4: Der Graph T und der erzeugte Prototyp P

5.4 Auswertung an den Testklassen

Für die Auswertung zur Bestimmung ähnlicher Teilgraphen sind keine quantitativen Vergleichsmerkmale bekannt. Daher wurde die Auswertung mit Clusterindizes

durchgeführt. Da die Ähnlichkeitsmatrix aber vollbesetzt ist und in Cliquen, d.h. vollständig verbundene Teilgraphen zerlegt wird, ist die Auswertung der einzelnen Partitionierungen P über komplizierte Clustering-Indizes nicht möglich. Indizes wie Performance, Conductance und Density führen hierbei nur auf triviale Ergebnisse. Daher kommt nur der *Coverage*-Index $\gamma(P)$ zum Einsatz, der auch die gewichteten Matrixeinträge, d.h. die Ähnlichkeitsmaße, berücksichtigt [BE05][Seite 180].

$$\gamma(P) = \frac{\sum\limits_{e \in P} e}{\sum\limits_{e \in E} e}$$

Allgemein kann erwartet werden, dass der Coverage-Index mit steigender Anzahl an Clustern kleiner wird, da der Schnitt mehr Kanten enthält. Anschließend müssen die einzelnen Cluster auf ihre Güte überprüft werden, d.h. ob bestimmte Teilgraphen zu geringe Ähnlichkeiten zu den anderen Teilgraphen des Clusters besitzen und dementsprechend gelöscht werden müssen oder ob Cluster aus einem Teilgraphen einem anderen Cluster hinzugefügt werden können. Dieses sogenannte *Refinement* der Partitionierung ist rechen- und zeitaufwendig.

5.4.1 Vergleich der Parameter an der Testklasse 150

Im folgenden Abschnitt sollen die aus der Anwendung der verschiedenen Verfahrensvarianten resultierenden Ergebnisse präsentiert werden. Die Verfahren wurden dabei an den 982 Graphen der Testklasse 150 verwendet.

Vergleich verschiedener Kostenfuntkionen

Für das Verfahren, das zur Bestimmung des Zentrums die Eigenwertzentralität verwendet, für die Ordnung der Nachbarschaft ebenfalls die Eigenwertzentralität nutzt und als zweites Merkmal die Knotenexzentrizitäten verwendet und die Ähnlichkeitsmatrix mit den Graph Edit-Distanzen aufstellt, soll die Tauglichkeit verschiedener Kostenfunktionen getestet werden. Die Kosten werden hierbei mit der Standardkostenfunktion und einer zweiten Kostenfunktion **CC2** berechnet. **CC2** nutzt dabei als zusätzliche Information, außer der Größe der Graphen, den durchschnittlichen Knotengrad ϑ

der beiden Graphen. Für die zweite Kostenfunktion gilt:

$$\alpha_G = \frac{1}{4 \times \max\{\min\{n(G), n(H)\}, |n(G) - n(H)|\} \times \vartheta(G)},$$

$$\alpha_H = \frac{1}{4 \times \max\{\min\{n(G), n(H)\}, |n(G) - n(H)|\} \times \vartheta(H)},$$

$$\beta_G = \frac{1}{8 \times \max\{\min\{m(G), n(H)\}, |n(G) - n(H)|\} \times \vartheta(G)},$$

$$\beta_H = \frac{1}{8 \times \max\{\min\{m(G), n(H)\}, |n(G) - n(H)|\} \times \vartheta(H)}.$$

Damit ergeben sich für beide Kostenfunktionen und die ersten 5 Reduktionsstufen (RS) der Ähnlichkeitsmatrizen, die Werte in Tab. 5.5. Die vielversprechends-

	GED_E_EM_CC1_D		GED_E_EM_CC2_D	
RS	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$
1	1	1	1	1
2	3	0.3107	6	0.2985
3	3	0.3073	28	0.0905
4	24	0.0548	75	0.0452
5	981	0.0000	157	0.0567

Tab. 5.5: Vergleich verschiedener Kostenfuntkionen $CC1$ und $CC2$ an der Testklasse 150

ten Partitionierungen ergeben sich für GED_E_EM_CC1_D bei $RS = 4$ und für GED_E_EM_CC2_D bei $RS = 3$. Beide Partitionen besitzen einen guten Coverageindex in Bezug auf die Anzahl an Clustern. Die Partition für GED_E_EM_CC2_D besitzt hierbei sogar einen besseren Coverageindex, trotz höherer Anzahl an Clustern. Betrachtet man die einzelnen Blöcke der Partition jedoch genauer, ergeben sich viele kleine Cluster aus ein oder zwei Knoten. Innerhalb der größeren Cluster ist die GED hoch. Die Umformungswege beinhalten jedoch durchschnittlich mehr Einfüge- und Löschoperationen, als Substitutionen. Dadurch wird die Erzeugung eines repräsentativen Prototypen erschwert.
So ergibt sich für den Cluster 5 der Prototyp aus Abb. 5.5 und die zugehörige Operationstabelle in Tab. 5.6. Diese Tabelle enthält die Größe des Teilgraphen T und die durchschnittlichen Größen der restlichen Teilgraphen des Clusters, sowie die durchschnittliche Anzahl der Umformungsoperationen von T zu den restlichen Teilgraphen.

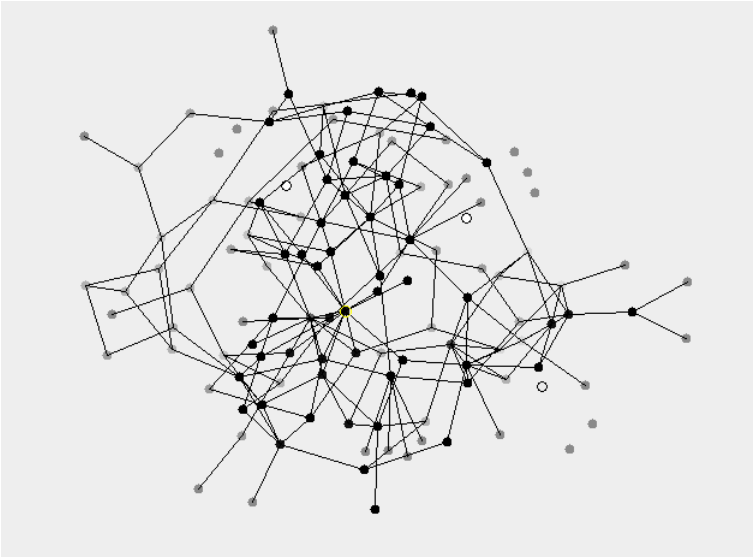

Abb. 5.5: Prototyp des Clusters 5 aus GED_E_EM_CC2_D mit $RS = 3$

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
34	124	210	115	196	50.42	73.58	80.85

Tab. 5.6: Operationstabelle des Clusters 5 aus GED_E_EM_CC2_D mit $RS = 3$

In Abb. 5.5 ist zu erkennen, dass bei der Erzeugung des Prototypen aus T Knoten gelöscht werden müssen, was vermieden werden soll.
Für GED_E_EM_CC1_D ergeben sich für $RS = 4$ ebenfalls viele kleine Cluster, die nicht zur Erzeugung von Prototypen genutzt werden können. Jedoch ist das Verhältnis der Umformungsoperationen innerhalb der Cluster im Vergleich besser. Als Beispiel dient der Cluster 13 (Abb. 5.6 und Tab. 5.7). Während der Erzeugung des

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
97	104	136	43	91	51.4	52.61	48.45

Tab. 5.7: Operationstabelle des Clusters 13 aus GED_E_EM_CC1_D mit $RS = 4$

Prototypen aus dem Teilgraphen T werden keine Knoten während der Generierung des Prototypen gelöscht.

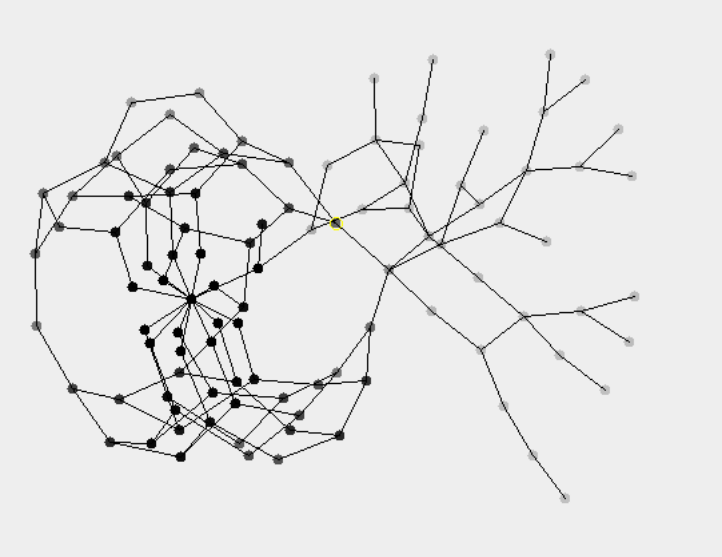

Abb. 5.6: Prototyp des Clusters 13 aus GED_E_EM_CC1_D mit $RS = 4$

Vergleich der Zentrumsbestimmung

Nun soll verglichen werden, wie sehr sich die Bestimmung des Zentrums, zu Beginn der Berechnung der suboptimalen GED, auf die Erzeugung der Prototypen auswirkt. Dazu werden drei Varianten miteinander verglichen, die das Zentrum jeweils mit der Eigenwertzentralität, der Knotenexzentrizität und dem Knotengrad bestimmen. Die Ordnung der Nachbarschaften sowie für die Kostenberechnung und Aufstellung der Ähnlichkeitsmatrix bleiben gleich.

	GED_E_EM_CC1_D		GED_M_EM_CC1_D		GED_D_EM_CC1_D	
RS	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$
1	1	1	1	1	1	1
2	3	0.3107	2	0.2753	2	0.3833
3	3	0.3073	8	0.0895	8	0.1168
4	24	0.0548	24	0.0393	40	0.0426
5	981	0.0000	78	0.0175	980	0.0000

Tab. 5.8: Vergleich verschiedener Zentrumsbestimumngen an der Testklasse 150

Die vielversprechendste Partition findet sich in Tab. 5.8 diesmal bei $RS = 4$. Die Variante GED_M_EM_CC1_D besitzt hierbei im Vergleich zu GED_E_EM_CC1_D einen geringeren Coverageindex bei gleicher Clusteranzahl. GED_D_EM_CC1_D besitzt jedoch sogar bei höherer Clusteranzahl einen besseren Coverageindex als GED_M_EM_CC1_D.

Für GED_M_EM_CC1_D mit $RS = 4$ ergeben sich für den Cluster 12 der Prototyp

aus Abb. 5.7, samt Operationstabelle Tab. 5.9.

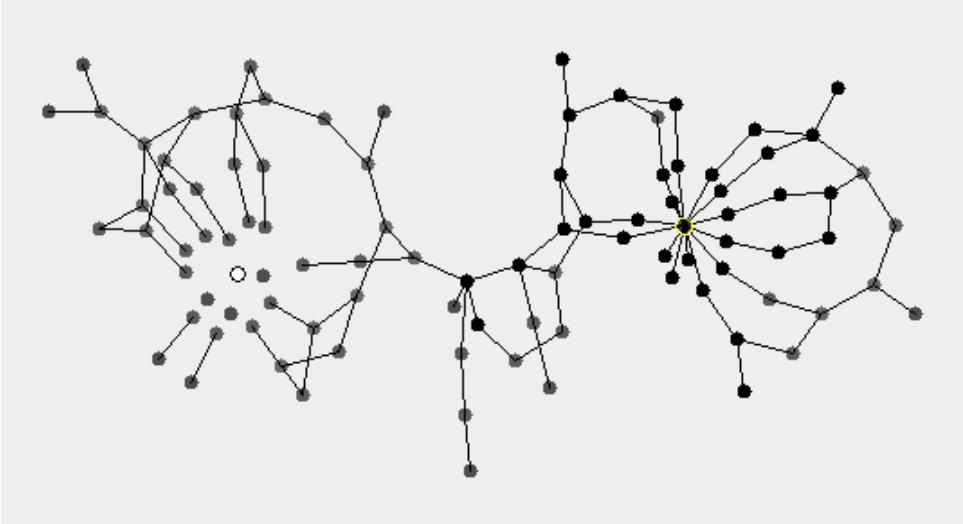

Abb. 5.7: Prototyp des Clusters 12 aus GED_D_EM_CC1_D mit $RS = 4$

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
91	96	115	63	102	32.75	63.26	73.57

Tab. 5.9: Operationstabelle des Clusters 12 aus GED_D_EM_CC1_D mit $RS = 4$

Auffällig ist dabei, dass der entstandene Prototyp nicht zusammenhängend ist, was ein Hinweis darauf ist, das die Umformungswege innerhalb des Clusters sehr verschieden sind. Daraus kann geschlossen werden, dass der Cluster noch weiter partitioniert werden kann, wenn Informationen der einzelnen Umformungswege genutzt werden und die Partitionierung gemäß den Umformungsoperationen geschieht.

Vergleich der Nachbarschaftsanordnung

Nachdem die Bestimmung des Zentrums verglichen wurde, sollen nun verschiedene Methoden zur Nachbarschaftsanordnung getestet werden. Dabei wird entweder die Einbettung in den $\mathbb{R}^2$ oder zwei Netzwerkzentralitäten verwendet. In letzterem Fall wird eine Netzwerkzentralität primär und eine sekundär, d.h. nur wenn die primäre nicht mehr ausreicht, genutzt. Dazu werden sechs Varianten miteinander verglichen, welche auch die Auswirkungen der Auswahl der sekundären Netzwerkzentralität betrachten soll. Die Bestimmung des Zentrums, die Kostenberechnung und Aufstellung

der Ähnlichkeitsmatrix bleiben hierbei gleich.

	GED_E_EM_CC1_D		GED_E_ED_CC1_D		GED_E_C_CC1_D	
RS	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$
1	1	1	1	1	2	0.4479
2	3	0.3107	3	0.3108	3	0.3646
3	3	0.3073	3	0.3072	14	0.0856
4	24	0.0548	25	0.0506	50	0.0391
5	981	0.0000	981	0.0000	146	0.00393

	GED_E_ME_CC1_D		GED_E_MD_CC1_D		GED_E_DE_CC1_D	
RS	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$
1	1	1	1	1	3	0.34449
2	3	0.3279	3	0.3204	3	0.3585
3	7	0.1462	3	0.3215	12	0.1044
4	41	0.0384	33	0.0441	55	0.0403
5	131	0.0297	981	0.00006	981	0.0000

Tab. 5.10: Vergleich verschiedener Nachbarschaftsanordnungen an der Testklasse 150

In Tab. 5.10 zeigt sich, dass die Wahl der sekundären Netzwerkzentralität geringen Einfluss auf den Coverageindex der Partitionierungen hat. GED_E_EM_CC1_D und GED_E_ED_CC1_D besitzen für die einzelnen Reduktionsstufen eine ähnliche Anzahl an Clustern und dazugehörigen Coverageindex. Gleiches gilt für GED_E_ME_CC1_D und GED_E_MD_CC1_D.
Die Verfahren GED_E_C_CC1_D und GED_E_DE_CC1_D führen hierbei bei $RS = 3$ bzw. $RS = 4$ zu einem, im Vergleich zu den anderen Verfahren, guten Coverageindex. Für $RS = 3$ ergaben sich jedoch aufgrund der geringeren Clusteranzahl ein schlechtes Operationsverhältnis innerhalb der größeren Cluster. So ergibt sich für GED_E_C_CC1_D mit $RS = 3$ der Cluster 3 mit folgender Operationstabelle:
Es werden innerhalb der Umformungswege deutlich mehr Knoten gelöscht bzw.

	$T(C)$		$\emptyset C$		Operationen		
$\lvert C\rvert$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
75	95	103	69	97	22.77	72.23	79.22

Tab. 5.11: Operationstabelle des Clusters 3 aus GED_E_C_CC1_D mit $RS = 3$

eingefügt, als substituiert. Ein aus einem solchen Cluster erzeugter Prototyp ist wenig repräsentativ.
Für $RS = 4$ ergeben sich jedoch die gleichen Probleme bei der Erzeugung der Prototypen, wie bei GED_D_EM_CC1_D. Das Verhältnis der Operationen im Cluster ist

gut (Tab. 5.10 und Tab. 5.11), wie Abb. 5.8 jedoch entnommen werden kann, sind die aus den Clustern der Partitionen erzeugten Prototypen nicht zusammenhängend.

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
21	95	147	111	180	50.01	44.92	72.22

Tab. 5.12: Operationstabelle des Clusters 1 aus GED_E_C_CC1_D mit $RS = 4$

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
51	111	199	98	194	54.02	56.98	67.94

Tab. 5.13: Operationstabelle des Clusters 3 aus GED_E_DE_CC1_D mit $RS = 4$

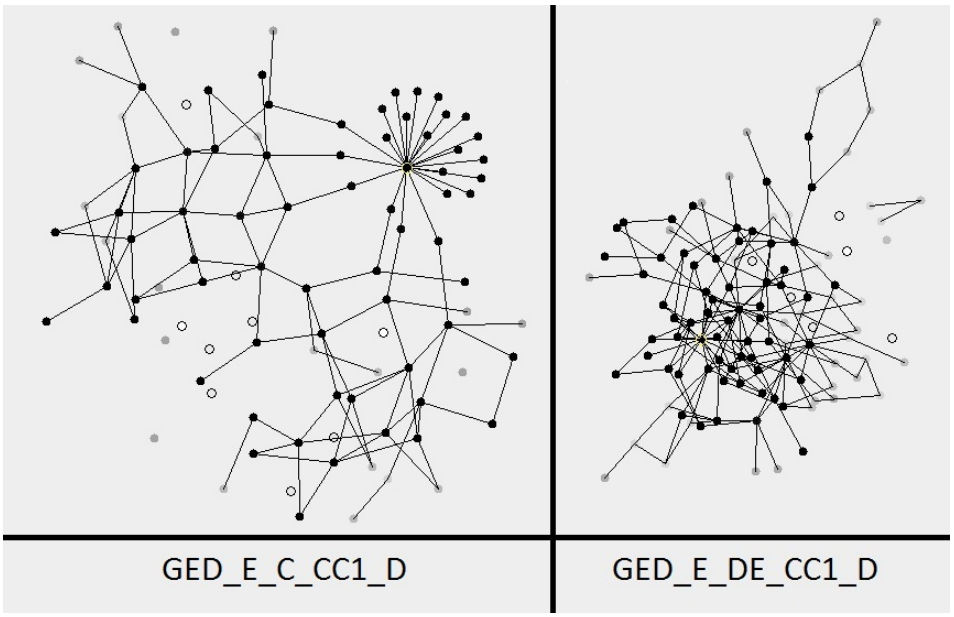

Abb. 5.8: Prototypen aus GED_E_C_CC1_D bzw. GED_E_DE_CC1_D mit $RS = 4$

Aufstellen der Ähnlichkeitsmatrix über den GEP

Nachdem die die Auswahl der verwendeten Kostenfunktion, der Bestimmung des Zentrums und der Nachbarschaftsordnung betrachtet wurden, soll nun der letzte Parameter der suboptimalen GED getestet werden. Dazu werden für Verfahrensvarianten aus [5.4.3] die Ähnlickeitsmatrizen mit Hilfe der Anzahl der Substitutionen aufgestellt. In Tab. 5.14 fällt auf, dass alle Ähnlichkeitsmatrizen bereits bei der ersten

	GED_E_EM_CC1_P		GED_E_C_CC1_P	
RS	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$
1	9	0.1662	15	0.1104
2	32	0.0712	53	0.0587
3	80	0.0547	112	0.0417
4	140	0.0458	196	0.0366
5	981	0.0000	266	0.0486
	GED_E_ME_CC1_P		GED_E_DE_CC1_P	
RS	Clusteranzahl	$\gamma(P)$	Clusteranzahl	$\gamma(P)$
1	8	0.1756	17	0.1010
2	40	0.0567	54	0.0745
3	93	0.0366	121	0.0363
4	168	0.0304	981	0.0000
5	252	0.043	265	0.0476

Tab. 5.14: Vergleich verschiedener Nachbarschaftsanordnungen an der Testklasse 150

Reduktionsstufe in mehrere Blöcke partitioniert werden können. Jedoch zerfallen die Ähnlichkeitsmatrizen bei $RS = 3$ schon in so viele Cluster, dass der Coverageindex darunter leidet. Daher werden nur die Partitionen mit $RS = 2$ betrachtet. Hier ergeben sich aus Tab. 5.12 die besten Coverageindizes für GED_E_DE_CC1_P bzw. GED_E_EM_CC1_P.

Innerhalb der Partitionierungen ergeben sich wieder zu kleine Cluster, deren Teilgraphen sich nicht ähnlich sind. Für GED_E_EM_CC1_P ergeben sich innerhalb der größeren Cluster gute Verhältnisse der Umformungsoperationen zueinander. Beispiel hierfür sei der Cluster 30. Aus Abb. 5.9 kann man erkennen, das bei der Erzeugung

	$T(C)$		$\emptyset C$		Operationen		
$\lvert C\rvert$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
58	95	153	98	165	51.58	43.42	67.19

Tab. 5.15: Operationstabelle des Clusters 30 aus GED_E_EM_CC1_P mit $RS = 2$

des Prototyps kein Knoten aus T gelöscht wird.

Für GED_E_DE_CC1_P ergibt sich nur ein durchschnittliches Verhältnis der Umformungsoperationen zueinander, die resultierenden Prototypen sind dennoch zusammenhängend und besitzen meist noch alle Knoten aus T, wie Tab. 5.14 und Abb. 5.10 zu entnehmen ist.

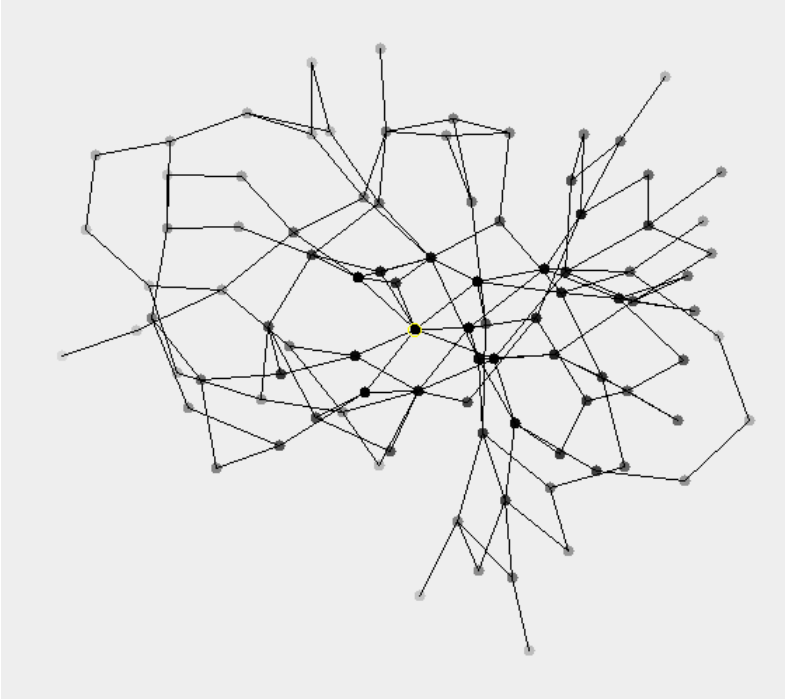

Abb. 5.9: Prototyp des Clusters 30 aus GED_E_EM_CC1_P mit $RS = 2$

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
73	137	186	75	149	47.51	89.59	66.96

Tab. 5.16: Operationstabelle des Clusters 18 aus GED_E_DE_CC1_P mit $RS = 2$

Schlussfolgerungen aus der Parameterauswertung

Beim Vergleich der verschiedenen Parametereinstellungen hat sich gezeigt, dass nicht alle Kombinationen zu verwertbaren Ergebnissen führen. Schon die richtige Auswahl, der zu betrachtenden Reduktionsstufe ist hierbei sehr wichtig. Da man die verschiedenen Partitionen bzw. die aus ihnen erzeugten Prototypen nicht miteinander vergleichen kann und alle Partitionierungen betrachtet werden müssten, wurde der Coverageindex γ zur Vorauswahl genutzt. Aus der Betrachtung der Partitionen hat sich ebenfalls gezeigt, dass eine zu geringe Clusteranzahl zu schlechten, teilweise nicht zusammenhängenden, Prototypen führt. Mit einer zu großen Anzahl an Clustern bzw. Clustern mit wenigen Teilgraphen kann jedoch kein wirklich *repräsentativer* Prototyp erzeugt werden. Diese Probleme ergeben sich jedoch für alle Verfahrensvarianten.
Bei der Aufstellung der Ähnlichkeitsmatrizen mit der GED tritt zusätzlich das Problem auf, dass der dem Zentrum am nächsten gelegene Teilgraph T relativ klein im

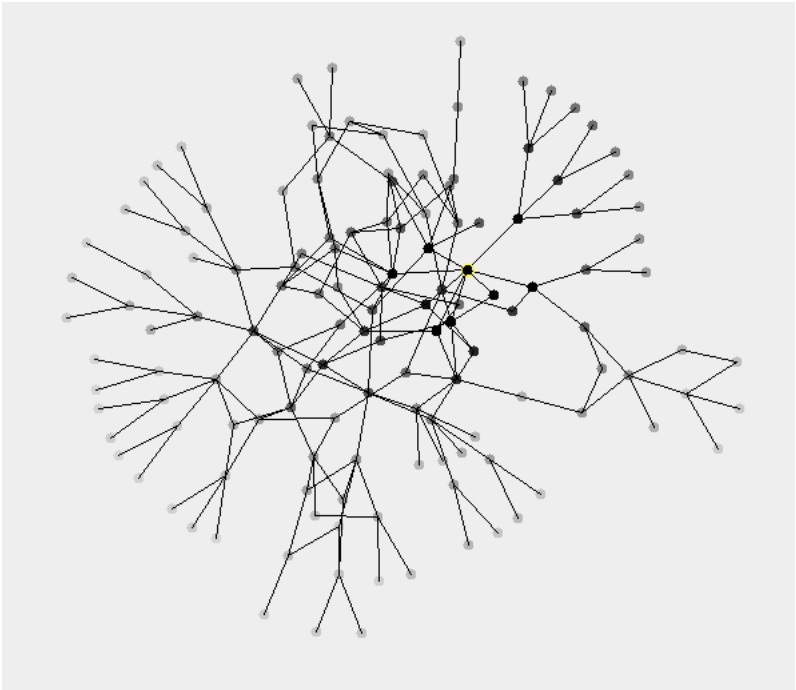

Abb. 5.10: Prototyp des Clusters 18 aus GED_E_EM_CC1_P mit $RS = 2$

Vergleich zu den restlichen Teilgraphen des Clusters ist. Dies ist auf die geringeren Kosten, die kleine Graphen bei der Berechnung der suboptimalen GED, erzeugen zurück zuführen. Da kleine Graphen weniger Knoten besitzen, werden ihre Nachbarschaften schneller gematcht, bevor die restlichen Knoten gelöscht bzw. eingefügt werden. Dadurch entfallen die teilweise mehrfachen Kosten für das Löschen/Einfügen innerhalb der Nachbarschaften für die Knoten [5.2.2]. Solche Verzerrungen wurden aufgehoben, indem die Ähnlichkeitsmatrizen direkt mit der Anzahl der Umformungsoperationen aufgestellt wurden. Bei den Clustern, die aus solchen Ähnlichkeitsmatrizen erzeugt wurden, gab es ebenfalls kaum Teilgraphen mit geringer Ähnlichkeit innerhalb der Cluster. Bei diesen Clustern muss kein Refinement mehr angewendet werden, um schwache Teilgraphen auszusortieren.

Weiterhin hat sich gezeigt, dass die besten Ergebnisse erzielt wurden, wenn die Eigenwertzentralität zur Bestimmung des Zentrums genutzt wurde. Für die Nachbarschaftsanordnung hat sich gezeigt, dass nur wenige brauchbare Kombinationen existieren bzw. dass das ursprüngliche Verfahren, basierend auf der Einbettung im $\mathbb{R}^2$, vernachlässigt werden kann.

Hierbei ergaben sich für die Testklasse 150 mit den Verfahren GED_E_EM_CC1_D und GED_E_DE_CC1_D bzw. GED_E_EM_CC1_P und GED_E_DE_CC1_P die besten

Ergebnisse. Im Folgenden werden nur die vier Varianten an den restlichen Klassen getestet.

5.4.2 Testklasse 300

Die vielversprechendste Partitionierung für die 504 Teilgraphen der Testklasse 300 findet sich in Tab. 5.17 für GED_E_EM_CC1_P mit $RS = 2$. Diese Partitionierung enthält wieder kleine Cluster, die aufgrund ihrer Größe eine geringere Grenze für die Anzahl der erlaubten Operationen für die Erzeugung eines Prototyps haben und damit nicht für die Generierung von Prototypen geeignet sind.

Jedoch zeigt sich hier deutlicher, als in der Testklasse 150, dass sich die Umformungswege innerhalb eines Clusters sehr unterscheiden können. So entstehen sogar aus Clustern mit keinen schwachen Teilgraphen, d.h. durchgehend hoher Ähnlichkeit, unzusammenhängende Prototypen, wie am Beispielcluster 10 in Abb. 5.11 zu erkennen ist.

	GED_E_EM_CC1_D		GED_E_DE_CC1_D	
RS	Clusteranzahl	Coverage	Clusteranzahl	Coverage
1	1	1	1	1
2	1	1	2	0.4530
3	2	0.3920	9	0.1270
4	13	0.0661	24	0.0579
5	33	0.0317	502	0.0000
	GED_E_EM_CC1_P		GED_E_DE_CC1_P	
RS	Clusteranzahl	Coverage	Clusteranzahl	Coverage
1	4	0.3056	8	0.2371
2	20	0.0876	38	0.0548
3	48	0.0384	72	0.0333
4	97	0.0351	133	0.0481
5	503	0.0000	503	0.0000

Tab. 5.17: Vergleich der GED für die Testklasse 300

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
45	217	305	163	234	65.86	151.14	120.48

Tab. 5.18: Operationstabelle des Clusters 10 aus GED_E_EM_CC1_P mit $RS = 4$ der Klasse 300

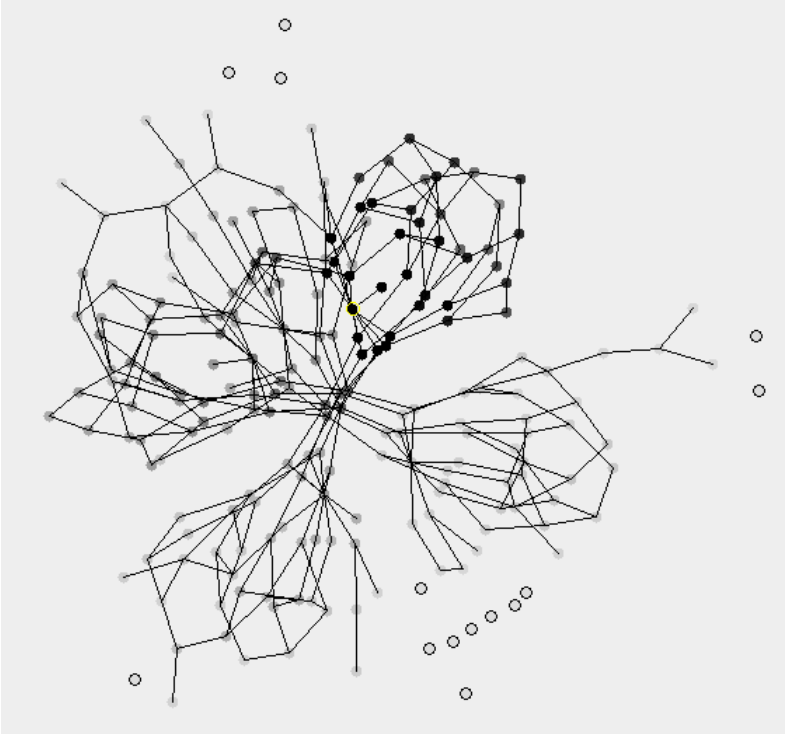

Abb. 5.11: Prototyp des Clusters 10 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 300

5.4.3 Testklasse 500

Die vielversprechendste Partitionierung in Tab. 5.19 für die 322 Teilgraphen der Testklasse 500 ergibt sich aus GED_E_EM_CC1_P mit $RS = 2$.

	GED_E_EM_CC1_D		GED_E_DE_CC1_D	
RS	Clusteranzahl	Coverage	Clusteranzahl	Coverage
1	1	1	1	1
2	2	0.2701	1	1
3	4	0.1866	7	0.1398
4	10	0.0778	321	0.0000
5	45	0.0283	318	0.0001
	GED_E_EM_CC1_P		GED_E_DE_CC1_P	
RS	Clusteranzahl	Coverage	Clusteranzahl	Coverage
1	9	0.1981	18	0.0972
2	21	0.1018	37	0.0580
3	49	0.0401	69	0.0350
4	76	0.0438	320	0.0001
5	122	0.0843	321	0.0000

Tab. 5.19: Vergleich der GED für die Testklasse 500

Es ist zu beobachten, dass ab einer bestimmten Größe der Teilgraphen T, die als Grundlage zur Erzeugung des Prototypen dienen, sich in bestimmten Prototypen selbst ähnlichen Teilgraphen erkennen lassen. Zur Verdeutlichung soll der Cluster 15 in Abb. 5.12 bzw. Tab. 5.20 dienen, der aus 4 ähnlichen Komponenten besteht.

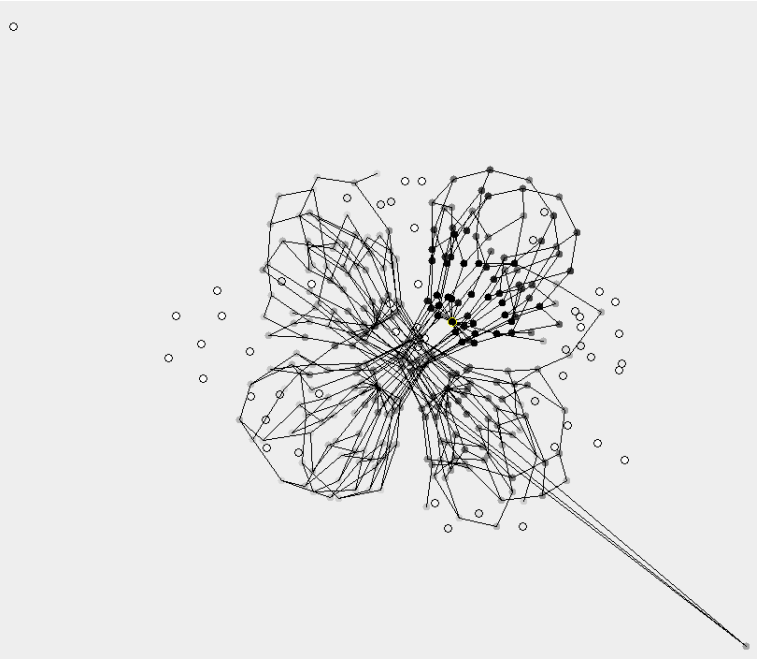

Abb. 5.12: Prototyp des Clusters 15 aus GED_E_EM_CC1_P mit $RS = 2$ der Testklasse 500

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
29	340	1510	285	407	137.29	202.71	162.07

Tab. 5.20: Operationstabelle des Clusters 15 aus GED_E_EM_CC1_P mit $RS = 2$ der Testklasse 500

5.4.4 Testklasse 700

Die vielversprechendste Partitionierung in Tab. 5.21 für die 220 Teilgraphen der Testklasse 700 ergibt sich wieder aus GED_E_EM_CC1_P mit $RS = 2$.

Für einige Cluster lassen sich selbst bei geringer Anzahl an Teilgraphen relativ gute Prototypen erzeugen, d.h. Prototypen, aus denen wenige Knoten gelöscht wurden.

	GED_E_EM_CC1_D		GED_E_DE_CC1_D	
RS	Clusteranzahl	Coverage	Clusteranzahl	Coverage
1	1	1	1	1
2	1	1	1	1
3	2	0.2621	11	0.0819
4	13	0.0650	43	0.0499
5	216	0.0002	217	0.0001
	GED_E_EM_CC1_P		GED_E_DE_CC1_P	
RS	Clusteranzahl	Coverage	Clusteranzahl	Coverage
1	6	0.2351	8	0.1988
2	22	0.0713	33	0.0533
3	40	0.0482	58	0.0445
4	215	0.0003	90	0.0812
5	218	0.0002	115	0.1318

Tab. 5.21: Vergleich der GED für die Testklasse 700

Als Beispiel soll der Cluster 4 mit nur 7 Teilgraphen in Abb. 5.13 und Tab. 5.22 dienen.

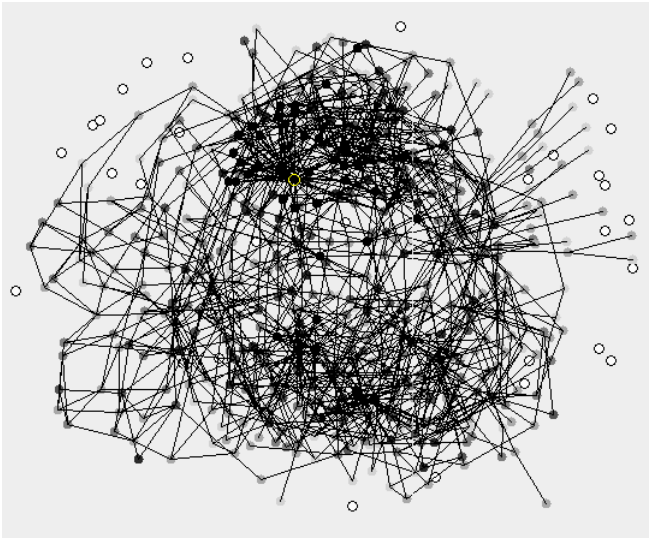

Abb. 5.13: Prototyp des Clusters 4 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
7	581	1282	541	1098	268.5	312.5	268.67

Tab. 5.22: Operationstabelle des Clusters 4 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 7000

Die bekannten Beobachtungen aus den Untersuchungen der anderen Testklassen zeigen sich für diese Klasse nun deutlicher. Da sich die Cluster aus größeren Teilgraphen zusammensetzen, enthalten manche Prototypen, aufgrund ihrer Größe, selbst wieder ähnliche Komponenten (Abb. 5.14 und Tab. 5.23) bzw. sind die Teilgraphen und damit die Umformungswege, innerhalb der Cluster so verschieden, dass keine guten Prototypen dabei entstehen (Abb. 5.15 und Tab. 5.24).

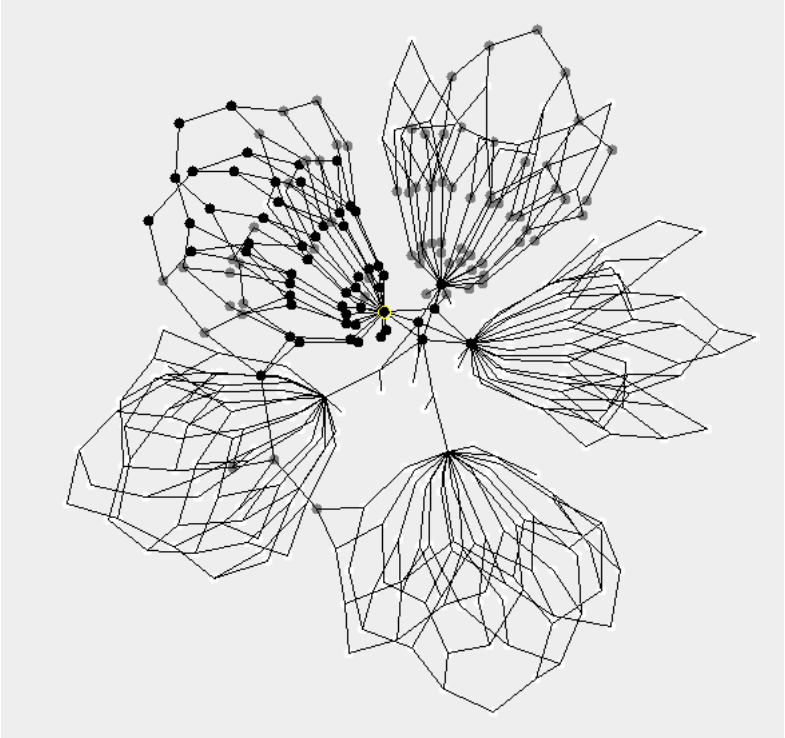

Abb. 5.14: Prototyp des Clusters 5 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
3	381	530	424	570	93	288	353.5

Tab. 5.23: Operationstabelle des Clusters 5 aus GED_E_EM_CC1_P mit $RS = 2$

	$T(C)$		$\emptyset C$		Operationen		
$\|C\|$	Knoten	Kanten	Knoten	Kanten	$\emptyset c_{ns}$	$\emptyset c_{nd}$	$\emptyset c_{ni}$
9	568	758	381	578	57.63	510.38	304

Tab. 5.24: Operationstabelle des Clusters 9 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700

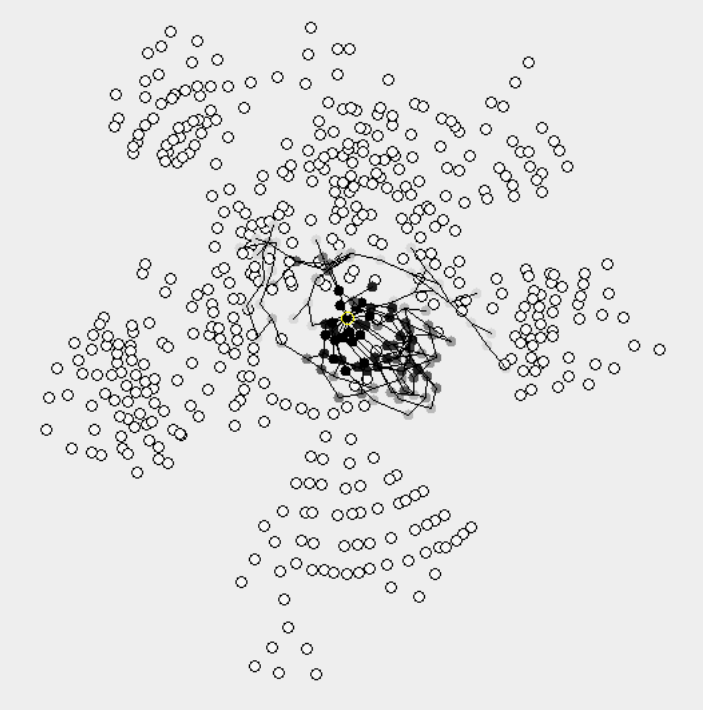

Abb. 5.15: Prototyp des Clusters 9 aus GED_E_EM_CC1_P mit $RS = 2$ der Klasse 700

5.5 Bemerkung zur Implementierung

Da die Adjazenzmatrizen der vorliegenden integrierten Schaltungen dünn besetzt sind, d.h. die Anzahl ihrer Nichtnullelemente signifikant kleiner als n^2 ist, wird im Folgenden das *Compressed Row Storage*-Format (CRS) eingeführt, um diese Eigenschaft auszunutzen.

Beim CRS-Format werden nur die von Null verschiedenen Einträge der Adjazenzmatrix A gespeichert. Dies geschieht in Form von vier Arrays. Der Array *obj_ptr* enthält die Label der Knoten aus A, *val* enthält die entsprechenden Werte der Nichtnullelemente aus A. In *col_ind* ist zu jedem Eintrag aus *val* der Index seiner Spalte gespeichert. Er besitzt die gleiche Länge wie *val*. Der Array *row_ptr* zeigt an, wie viele Spalteneinträge eine Zeile aus A besitzt.

Um eine Matrix A im CRS-Format darstellen zu können, wird folgender formaler Zusammenhang genutzt:

$$A_{i,j} = val[k] \Leftrightarrow (col\_ind[k] = j) \wedge (row\_ptr[i] \leq k < row\_ptr[i+1]).$$

Beispiel für das CRS-Format:

Gegeben sei die Adjazenzmatrix A, deren Spalten bzw. Zeilen gleich der Label ihrer Knoten ist.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \xrightarrow{\text{im CRS-Format}} \begin{aligned} obj\_ptr &= \{1, 2, 3, 4\} \\ val &= \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\} \\ col\_ind &= \{2, 3, 1, 3, 4, 1, 2, 4, 2, 3\} \\ row\_ptr &= \{0, 2, 5, 8, 10\} \end{aligned}$$

6 Zusammenfassung und Ausblick

Ziel dieser Diplomarbeit war es, integrierte Schaltungen auf gleiche Komponenten zu überprüfen. Dies wurde durch ein mehrstufiges Verfahren realisiert, welches in Java implementiert wurde. Erster Schritt dieses Verfahrens ist das Zerlegen der Schaltung in vergleichbare Komponenten. Hierfür wurden spektrale Partitionierungsalgorithmen vorgestellt und angewandt. Bei der Partitionierung der Schaltung ist jedoch aufgefallen, dass die spektralen Verfahren nur begrenzt anwendbar sind. Für große dünnbesetze Matrizen ergeben sich zwar keine hohen Rechenzeiten, aber vor allem die Eigengap-Heuristik, einer der ausschlaggebenden Gründe bei der Wahl dieses Partitionierungsverfahrens, hat sich als kaum anwendbar für das hierarchische Partitionieren eines Graphen herausgestellt. Der Eigengap liegt bei vielen Graphen genau bei $k = 1$, d.h. der Graph ist durch den Eigengap nicht weiter zerlegbar. Auch die Verwendung des zweitgrößten Eigengaps hat sich als praktisch kaum anwendbar herausgestellt. Hier sollte in Zukunft ein Verfahren angewandt werden, welches bei unbrauchbarem Eigengap den Graphen iterativ mit spektraler Multisektion bei steigendem k zerlegt, bis eine verwertbare Partitionierung gefunden wurde.
Das Zerlegen in zwei gleich große Teilgraphen mittels spektraler Bisektion ist praktisch ebenfalls nicht anwendbar, da die daraus entstehenden Teilgraphen teilweise nicht zusammenhängend sind. Erst die Verwendung der spektralen Multisektion mit festem k führt zu angemessenen Ergebnissen. Hierbei fehlt jedoch die Kontrolle über die Größe und Anzahl der entstehenden Teilgraphen.
Hier sollten in Zukunft andere Partitionierungsverfahren und -strategien angewendet werden, die für große Graphen geeignet sind.
Weiterhin wurde innerhalb der Arbeit das Ähnlichkeitsmaß der Graph Edit-Distanz eingeführt, die es ermöglicht, Graphen nicht nur auf Ähnlichkeit miteinander zu vergleichen, sondern gelichzeitig einen Umformungsweg zwischen Graphen erzeugt. Ebenfalls wurde ein suboptimaler Algorithmus zur Berechnung der GED vorgestellt,

der auf der Einbettung der Graphen im $\mathbb{R}^2$ beruht. Mit Hilfe der Netzwerkzentralitäten wurden alternative Varianten zur suboptimalen Berechnung der GED aufgezeigt und anschließend miteinander auf ihre Tauglichkeit verglichen. Die Einbettung in den $\mathbb{R}^2$ und die Berechnung der Netzwerkzentralitäten der Graphen können parallelisiert werden. Die vielversprechendste Berechnungsvariante basiert dabei auf die Nutzung der Eigenwertzentralität zur Erkennung des Zentrums des Graphen und zur Ordnung der Zeichenkette, sowie auf der Erzeugung der Ähnlichkeitsmatrix über die Anzahl der Umformungsoperationen.
Weitere Varianten zur Bestimmung der suboptimalen GED könnten folgende Ansätze weiterverfolgen:

- Anwendung auf gerichtete Graphen → mehrmalige Anwendung der Zentrumsfindung, wenn Knoten in beiden Graphen übrig sind, welche noch nicht substituiert wurden und die Anwendung auf gelabelte Graphen. Dadurch soll das Hintergrundwissen über die Zellen der integrierten Schaltungen zur Verbesserung des Ähnlichkeitsmaßes genutzt werden. Auch die Nutzung anderer Netzwerkzentralitäten ist damit möglich.

- Bei mehreren Durchläufen sollen die Nachbarschaftsknoten des Zentrums als Startpunkte dienen. Dadurch soll eine Verbesserung der Robustheit des Algorithmus erreicht werden.

- Testen weiterer Kostenfunktionen (z.B. auf Grundlage der Netzwerkzentralitäten).

- Entwicklung eines Algorithmus, der weitere strukturelle Eigenschaften (z.B. Kreise) des Graphen unter Einbezug weiterer Graphenoperationen, z.B. der Knotenkontraktion und -fusion, berücksichtigt. Auch das Auftauchen ähnlicher Komponenten innerhalb eines Prototyps sollte weiter betrachtet werden.

- Berechnung der suboptimalen GED mit Hilfe spektraler Methoden, welche die gesamte Adjazenzmatrix in eine Zeichenkette umwandeln [RkH05].

Ebenso zeigt sich, dass Verfahren, welche zur Erstellung der Ähnlichkeitsmatrizen die GEP nutzen, zu besseren Ergebnissen führen, als Verfahren, welche dafür nur die GED nutzen. Die Partitionierung der Ähnlichkeitsmatrix mit der spektralen Multisektion gibt bei entsprechender Reduzierung der Matrixeinträge gute Partitionierungen

zurück. Eine raffiniertere Auswertung der Ähnlichkeitsmatrix, u.a. mit anderen Clusterverfahren, könnte weiterhin zur Verbesserung des Verfahrens beitragen.
Auch die Bestimmung der Prototypen aus den Clustern durch die Graph Edit-Paths sollte nicht nur mit Hilfe der zu löschenden Knoten durchgeführt werden. Es sollten auch die einzufügenden Knoten bestimmt werden. Dazu muss jedoch eine Verfahren zum Vergleich mehrerer Graphen entwickelt werden und damit auch zum Vergleich der GEP mehrerer Graphen. Dies wäre jedoch mit hohem Speicher- und Laufzeitaufwand verbunden. Dadurch könnten die einzelnen Umformungsoperationen aber aussagekräftiger zur Erstellung der Prototypen genutzt werden. Basierend auf der Erzeugung von Prototypen aus ähnlichen Grundstrukturen kann eine Datenbank von Prototypen angelegt werden, mit deren Hilfe weitere integrierte Schaltungen untersucht werden.

Literaturverzeichnis

[Alb07] Albers, Jan: *Grundlagen integrierter Schaltungen - Bauelemente und Mikrostrukturierung*. Carl Hanser Verlag München, 2007

[Ant71] Anthonisse, J.M.: The rush in a directed graph., Technical Report BN9/71 / Stichting Mahtematisch Centrum, Amsterdam. 1971. – Forschungsbericht

[ANTT02] Arasu, Arvind ; Novak, Jasmine ; Tomkins, Andrew ; Tomlin, John: PageRank Computation and the Structure of the Web: Experiments and Algorithms. In: *Proceedings of the Eleventh International World Wide*, 2002

[BB93] Bunke, H. ; Buhler, U.: Applications of approximate string matching to 2D shape recognition. In: *Pattern Recognition* (1993)

[BE05] Brandes, Ulrik ; Erlebach, Thomas: *Network Analysis: Methodological Foundations (Lecture Notes in Computer Science)*. Springer-Verlag New York, 2005

[Ber89] Berge, Claude: *Hypergraphs*. Bd. 45. North-Holland Mathemtical Library, 1989

[BKL08] Bunke, Horst (Hrsg.) ; Kandel, Abraham (Hrsg.) ; Last, Mark (Hrsg.): *Studies in Computational Intelligence*. Bd. 91: *Applied Pattern Recognition*. Springer, 2008. – ISBN 978–3–540–76830–2

[BN07] Bunke, Horst ; Neuhaus, Michel: *Series in Machine Perception and Artificial Intelligence*. Bd. 68: *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific Publishing Co., Inc. River Edge, NJ, USA, 2007

[Bra01] Brandes, Ulrik: A Faster Algorithm for Betweenness Centrality. In: *Journal of Mathematical Sociology* 25 (2001), S. 163–177

[BS98] Bunke, Horst ; Shearer, Kim: *A Graph Distance Metric Based on the Maximal Common Subgraph.* 1998

[Bun99] Bunke, Horst: Error correcting graph matching: on the influence of the underlying cost function. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21 (1999), sep, Nr. 9, S. 917–922

[BVD04] Blondel ; Van Dooren, et a.: A measure of similarity between graph vertices. With Apllications to Synonym extraction and web searching. In: *SIAM Review* (2004)

[Fie73] Fiedler, M.: Algebraic connectivities of graphs. In: *Czech. Math. J.* 23 (1973), S. 298–305

[Fre77] Freeman, Linton C.: *A Set of Measures of Centrality Based on Betweenness, Sociometry 40.* 1977. – 35-41

[HK92] Hagen, Lars W. ; Kahng, Andrew B.: New spectral methods for ratio cut partitioning and clustering. In: *IEEE Trans. on CAD of Integrated Circuits and Systems* (1992), S. 1074–1085

[Jol01] Jolion, J.M.: *Graph Matching: What Are We Really Talking About?* 2001

[JW02] Jeh, Glen ; Widom, Jennifer: SimRank: A Measure of Structural-Context Similarity. In: *In KDD*, 2002, S. 538–543

[KK89] Kamada, T. ; Kawai, S.: An algorithm for drawing general undirected graphs. / Information Processing Letters. 1989 (31). – Forschungsbericht. – 7-15

[Kle99] Kleinberg, J. M.: Authoritative sources in a hyperlinekd enviroment. In: *J.ACM* 46 (1999), S. 604–632

[KMN+02] Kanungo, Tapas ; Mount, David M. ; Netanyahu, Nathan S. ; Piatko, Christine D. ; Silverman, Ruth ; Wu, Angela Y. ; Member,

Senior ; MEMBER, Senior: An efficient k-means clustering algorithm: Analysis and implementation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), S. 881–892

[Lev65] LEVENSHTEIN, Vladimir I.: *Binary codes capable of correcting deletions, insertions, and reversals.* 1965. – 845–848 S. – Russisch, Englische Übersetzung in: Soviet Physics Doklady, 10(8) S. 707?710, 1966

[Lux07] LUXBURG, Ulrike: A tutorial on spectral clustering. In: *Statistics and Computing* 17 (2007), December, S. 395–416

[MB98] MESSMER, B.T. ; BUNKE, H.: A new algorithm for error-tolerant subgraph isomorphism detection. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20 (1998), may, Nr. 5, S. 493–504

[MGmR02] MELNIK, Sergey ; GARCIA-MOLINA, Hector ; RAHM, Erhard: Similarity flooding: A versatile graph matching algorithm, 2002, S. 117–128

[MJ97] MOHAR, Bojan ; JUVAN, Notes Taken M.: Some Applications of Laplace Eigenvalues of Graphs. In: *Graph Symmetry: Algebraic Methods and Applications, volume 497 of NATO ASI Series C*, Kluwer, 1997, S. 227–275

[Neu75] NEUMANN, Klaus: *Operations Research Verfahren - Band 1: Lineare Optimierung, Spieltheorie, Nichtlineare Opitmierung, Ganzzahlige Optimierung.* Carl Hanser Verlag München Wien, 1975

[New10] NEWMAN, M.E.J.: *Networks: An Introduction.* Oxford University Press, 2010

[PM99] PAPADOPOULOS, A. N. ; MANOLOPOULOS, Y.: Structure-Based Similarity Search with Graph Histograms. In: *Proceedings of International Workshop on Similarity Search* (1999), S. 174–178

[PSL90] POTHEN, A. ; SIMON, H. ; LIOU, K-P: Partitioning sparse matrices with Eigenvectors of graphs. In: *SIAM J. Matrix Appl.* 11 (1990), S. 430–452

[RkH05] Robles-kelly, Antonio ; Hancock, Edwin R.: Graph edit distance from spectral seriation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), S. 365–378

[Saa03] Saad, Y.: *Iterative Methods for Sparse Linear Systems*. SIAM Society for Industrial & Applied Mathematics, 2003

[Sch04] Schönauer, Stefan: *Efficient Similarity Search in Structured Data.*, LMU München: Fakultät für Mathematik, Informatik und Statistik, Diss., 2004

[Sch07] Schaeffer, Satu E.: Graph clustering. In: *Computer Science Review* 1 (2007), Nr. 1, S. 27–64

[SM00] Shi, Jianbo ; Malik, Jitendra: Normalized cuts and image segmentation. 2000. – Forschungsbericht

[Tit03] Tittman, P.: *Graphentheorie, Eine anwendungsorientierte Einführung*. Carl Hanser Verlag München Wien, 2003

[WF74] Wagner, R. ; Fischer, M.: The string-to-string correction problem. In: *Journal of the ACM* (1974), Nr. 21(1), S. 168–173

[WM03] Washio, Takashi ; Motoda, Hiroshi: State of the art of graph-based data mining. In: *SIGKDD Explorations* 5 (2003), Nr. 1, S. 59–68

[YH02] Yan, Xifeng ; Han, Jiawei: gSpan: Graph-Based Substructure Pattern Mining. In: *Proceedings of the 2002 IEEE International Conference on Data Mining*. Washington, DC, USA : IEEE Computer Society, 2002 (ICDM '02), 721–

[Zag05] Zager, Laura: *Graph similarity and matching*, Massachusetts Institute of Technology, Diplomarbeit, 2005

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.
Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Dresden, 19. Oktober 2011

Enrico Goldhan